Karl Erich Wolff
Heather D. Pfeiffer
Harry S. Delugach (Eds.)

LNAI 3127

Conceptual Structures at Work

12th International Conference on Conceptual Structures, ICCS 2004
Huntsville, AL, USA, July 2004
Proceedings

Springer

Lecture Notes in Artificial Intelligence 3127

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

This page intentionally left blank

Karl Erich Wolff Heather D. Pfeiffer
Harry S. Delugach (Eds.)

Conceptual Structures at Work

12th International Conference
on Conceptual Structures, ICCS 2004
Huntsville, AL, USA, July 19-23, 2004
Proceedings

Springer

eBook ISBN: 3-540-27769-2
Print ISBN: 3-540-22392-4

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

This volume contains selected papers presented at the 12th International Conference on Conceptual Structures, ICCS 2004, held in Huntsville Alabama, July 19–23, 2004.

The main theme of the conference, "Conceptual Structures at Work", was chosen to express our intention of applying conceptual structures for human-centered practical purposes. That invites us to develop not only clear conceptual theories, but also methods to support humans in the application of these theories in their societies. Some promising steps in this direction are being taken, but the gap between the researchers working on a highly sophisticated level on one side and the practitioners in many fields of applications on the other side is usually difficult to bridge. Some of us have experiences in such practical cooperation, but we need more members of our community to be engaged in "real life problems".

We all know that solutions of complex problems in practice require not only a well-developed formal theory, but also an understanding of the whole context of the given problems. To support our understanding we need general philosophical methods as well as formal theories for the representation of fundamental structures in practice. We believe that our community has powerful tools and methods for successful applications in practice, but that we must develop a forum to present our results to a broader audience. First we must understand the significant developments in our own group, which has activities in many directions of research.

For that purpose, we have invited five distinguished researchers, covering many aspects of conceptual structures and methods, to present their views on fundamental problems in conceptual knowledge processing. These talks are presented in the first five papers of this volume. We express our gratitude to the invited speakers for the substantial and clear presentation of their results.

The papers in this volume cover the most important areas of research in the field of conceptual structures: from philosophical foundations over logical investigations to conceptual graphs, and further on to conceptual frameworks in practical applications and reasoning with conceptual structures.

All the papers in this volume were reviewed by at least two reviewers belonging to the Program Committee and by one member of the Editorial Board. We thank the authors for their submissions, and express our gratitude to the members of the Program Committee and the Editorial Board for their worthy support of our research community in the field of conceptual structures.

July 2004

Karl Erich Wolff
Heather D. Pfeiffer
Harry Delugach

This page intentionally left blank

Organization

The International Conference on Conceptual Structures (ICCS) is the annual conference and the principal research forum in the theory and practice of conceptual structures. Previous ICCS conferences were held at the Université Laval (Quebec City, 1993), at the University of Maryland (1994), at the University of California (Santa Cruz, 1995), in Sydney (1996), at the University of Washington (Seattle, 1997), in Montpellier (1998), at Virginia Tech (Blacksburg, 1999), at Darmstadt University of Technology (2000), at Stanford University (2001), at Borovets, Bulgaria (2002), and at Dresden University of Technology (2003).

General Chair

Harry Delugach	University of Alabama in Huntsville, USA

Program Chairs

Heather D. Pfeiffer	New Mexico State University, USA
Karl Erich Wolff	Darmstadt University of Applied Sciences, Germany

Editorial Board

Program Committee

Anne Berry (France)
Tru Cao (Vietnam)
Frithjof Dau (Germany)
Pavlin Dobrev (Bulgaria)
John Esch (USA)
David Genest (France)
Ollivier Haemmerlé (France)
Roger Hartley (USA)
Udo Hebisch (Germany)
Joachim Hereth Correia (Germany)
Richard Hill (UK)
Kees Hoede (The Netherlands)
Julia Klinger (Germany)
Pavel Kocura (UK)
Leonhard Kwuida (Germany)
Carsten Lutz (Germany)
Philippe Martin (Australia)
Sergei Obiedkov (Russia)
Peter Øhrstrøm (Denmark)
Simon Polovina (UK)
Anne-Marie Rassinoux (Switzerland)
Gary Richmond (USA)
Olivier Ridoux (France)
Dan Rochowiak (USA)
Sebastian Rudolph (Germany)
Eric Salvat (France)
Janos Sarbo (The Netherlands)
Henrik Schärfe (Denmark)
Thanwadee T. Sunetnanta (Thailand)
Petko Valtchev (Canada)

Table of Contents

Invited Talks

From Data to Conceptual Structures

Concept Lattices and Concept Graphs

Conceptual Frameworks for Applications

Reasoning with Conceptual Structures

This page intentionally left blank

The Esthetic Grounding of Ordered Thought

Kelly Parker

Department of Philosophy
Grand Valley State University
Allendale, MI 49401, USA
parkerk@gvsu.edu

Abstract. In Peirce's account of the normative sciences, logical validity and truth ultimately rely upon – though they are not reduced to – ethical and esthetic insight. An examination of the relations among critical esthetics, ethics and logic in Peirce's system suggests a possible account of the logic of creative discovery, which Peirce identified as abductive inference.

1 Introduction

In 1898 Charles S. Peirce delivered a series of lectures in Cambridge, Massachusetts under the title "Reasoning and the Logic of Things," in which he devoted considerable attention to the circumstances that motivate and structure our thinking. Several characteristic themes emerged as Peirce developed this line of thought in subsequent years. First, inquiry and knowledge are primarily community enterprises: while individuals do the work of observation and experimentation, the meaning of their results only arises within the community of inquirers. Second, Peirce favors metaphysical idealism over reductive materialism. Laws, which are represented as general ideas in the signs used by communities, determine the ordered relations that constitute reality. Third, he favors epistemological realism over nominalism. The truth of ideas is not a matter of mere convention or the psychological dispositions of sign-users. Rather, true ideas represent an objective order of the world. Fourth, the logic of successful inquiry presupposes a certain ethical orientation of selfless devotion to furthering the community's knowledge. Finally, there is an esthetic insight that motivates and grounds inquiry. This esthetic insight is even more fundamental than the inquirers' ethical orientation and in fact would seem to be at the very basis of thought itself. Since the world itself – or at least the world as we can ever hope to know it at all – consists in thought, this amounts to saying that the most basic fact of the knowable world is an esthetic fact of some kind.

Although Peirce's writings are notoriously dispersed and fragmentary, and his writings on esthetics and ethics are the least developed of all his works, it is clear that esthetic and ethical insight play a definitive role in inquiry as Peirce conceived it.[1] By 1903 Peirce had developed a broad notion of logic as semeiotic, the method of a community of inquirers committed to truth-seeking. Members of this community embrace the principle of fallibilism and maintain the hope that there is a reality their logic and language are capable of representing. Peirce came to believe that inquiry requires a definite esthetic

[1] For an extensive list of relevant materials see [7].

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 1–16, 2004.

insight into what is good and admirable. Knowledge is not at all a matter of "neutral observation." Esthetic insights motivate our interests and decisively influence our choice of ordering systems.

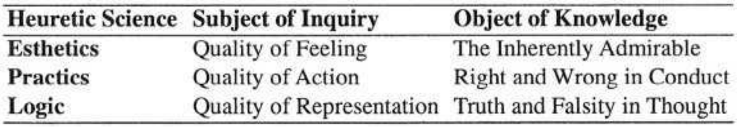
There are several alternative accounts of the basis of ordered thought in contemporary philosophy of science. The older view, held by researchers from Plato to the Encyclopedists, maintained that there is one pre-existing structure of knowledge waiting to be discovered. Peirce, Dewey, James and other pragmatists famously challenged this view at the turn of the nineteenth century. More recently, Thomas Kuhn detailed how social and political factors, as well as personal interests, have a significant effect on what knowledge is discovered and how it is ordered and presented [6]. The pressing question now concerns the extent to which factors such as emotion, esthetic appeal, morality, social pressure, political and economic power, and commercial interests legitimately affect the structures of knowledge. On one hand, Karl Popper argues that the formation of knowledge is governed by logical procedures and principles – given a certain context of accepted background principles. On the other hand, some post-modern social constructivist thinkers would maintain that knowledge is largely or entirely determined by contingent, irrationalist factors.

While the claim that interest influences the structure of knowledge could hardly shock anyone familiar with post-Kuhnian philosophy of science, Peirce adds a notable stipulation that situates him between these extremes: objective judgment and criticism can be brought to bear on the non-rational and contingent insights that influence the construction of knowledge. The relationship among esthetic, ethical, and logical judgment is developed in Peirce's accounts of the normative sciences. Attention to these relationships suggests that all three are essential elements in a logic of creativity and discovery.

2 Peirce's Value Theory: The Normative Sciences

The three normative sciences appear at the center of Peirce's mature classification of the sciences, which outlines his system of thought. In this architectonic classification, mathematics and mathematical logic appear as the first major division. Peirce conceives mathematics as the purely hypothetical investigation of what conclusions follow from arbitrarily adopted postulates. The next major division is phenomenology, the nearly passive observation of the structures of experience. After phenomenology comes esthetics, the first normative science. Esthetics is the science of ideals: its aim is to formulate a concept of the *summum bonum,* that which is admirable in itself. The second normative science is practics, the inquiry into the nature of right and wrong action. The last of the normative sciences is logic, or semeiotic, which investigates the principles of the representation of truth. The studies preceding the normative sciences in the system do virtually nothing to affect reality, while those that follow – beginning with metaphysics and including the special sciences and practical arts – are increasingly directed toward understanding and altering reality in various ways. The normative sciences give these latter activities their direction: it is in normative science that we critically examine the ends that guide our interactions with the world, including the action of knowing the world and embodying that knowledge in usable orders. Thus, for Peirce, questions of value precede not only action, but they also precede most questions of fact (excepting only the most general questions of formal fact concerning mathematical relations, and those concerning the structure of experience which are raised in phenomenology). Peirce's system thus embraces the post-Enlightenment idea that "all facts are value-laden." Because the values in question are themselves the products of methodological inquiry, though, this system rejects the notion – advanced, for example, by Michel Foucault – that all the facts are ultimately arbitrary constructs.

Table 1. The Normative Sciences



Heuretic Science	Subject of Inquiry	Object of Knowledge
Esthetics	Quality of Feeling	The Inherently Admirable
Practics	Quality of Action	Right and Wrong in Conduct
Logic	Quality of Representation	Truth and Falsity in Thought

The normative sciences each address a particular mode of interaction with the world (Table 1): "For Normative Science in general being the science of the laws of conformity of things to ends, esthetics considers those things whose ends are to embody qualities of **feeling,** ethics those things whose ends lie in **action,** and logic those things whose end is to **represent** something" [CP, 5.129], [, 2:200,].

There is moreover a clear interdependence among the three normative sciences. Each subsequent science considers a kind of end that is a narrower aspect of its predecessor's focus. "Logical goodness and badness, which we shall find is simply the distinction of *Truth* and *Falsity* in general, amounts, in the last analysis, to nothing but a particular application of the more general distinction of Moral Goodness and Badness, or Righteousness and Wickedness" [CP, 5.108], [EP, 2:188]. In turn, "if the distinction [between] Good and Bad Logic is a special case [of the distinction between] Good and Bad Morals, by the same token the distinction of Good and Bad Morals is a special case of the distinction [between] esthetic Goodness and Badness" [CP, 5.110], [EP, 2:189]. In general, the three normative sciences "may be regarded as being the sciences of the conditions of truth and falsity, of wise and foolish conduct, of attractive and repulsive ideas" [CP, 5.551], [EP, 2:378]. Peirce maintains, in short, that Truth is a species of the Right, which is in turn a species of the Admirable in general [CP, 5.130], [EP, 2:201].

Thought or representation ought to conform to its proper ideal, which is just to say that we strive for our thoughts to be true. Logic in the broadest sense is the study of the conditions under which thought can reliably be considered to conform to Truth. Likewise, action has its own proper ideal. Any action that is Right conforms to this ideal (whatever the ideal may be found to involve). It is the concern of Practics to articulate the conditions under which action can reliably be considered to conform to what is Right. So far Peirce was traveling familiar territory. When we come to esthetics, however, we find him departing from traditional conceptions of esthetics as the study of Beauty, or of the Pleasant.

2.1 Esthetics

Peircean esthetics is explained in terms exactly analogous to logic and practics: the concern of esthetics is to articulate the conditions under which our feelings can reliably be considered to conform to the Admirable.[2] In esthetics we enter a shadowy realm of gradations of feeling: “that dualism which is so much marked in the True and False, logic’s object of study, and in the Useful and Pernicious of the confessional of Practics, is softened almost to obliteration in esthetics” [CP, 5.551], [EP, 2:379]. Esthetics aims to distinguish the “nobility” of feelings, which is a matter of how well they conform to the standard of the *summum bonum*.[3] As a normative science, its role is to describe the basis of *admirable feeling*. In Peirce’s system, admirable feeling is the foundation upon which decisive action and critical thought mount their own more specific ideals:

> If conduct is to be thoroughly deliberate, the ideal [that guides it] must be a habit of feeling which has grown up under the influence of a course of self-criticisms and hetero-criticisms; and the theory of the deliberate formation of such habits of feeling is what ought to be meant by *esthetics*. [CP, 1.574].

As a normative science, esthetics proceeds upon the principle that habits of feeling may be as deliberately shaped as habits of action or thought. They may be cultivated to better conform to the Admirable, just as action may be deliberately cultivated to conform to the Right and thought to the True. Once habits of feeling that favor the Admirable are established, the ground is laid for good actions and thinking to follow more naturally.

All of this presumes that there is a singular standard or ideal that can be identified among the vast panorama of things that people actually do find attractive. Such an ideal must, on Peirce’s view, recommend itself on pre-experiential grounds. That is, it must be such that it is uniquely suited as an object of feeling. The question of esthetics, accordingly, is “What is the one quality that is, in its immediate presence, καλος?" [CP, 2.199]. What state of things is admirable in itself? Peirce tentatively answers: “an object, to be esthetically good, must have a multitude of parts so related to one another as to impart a simple positive quality to their totality,” *whatever* that quality may be [CP, 5.132], [EP, 2:201]. When applied to the totality of all that is, the evolving universe, the *summum bonum* consists “in that process of evolution whereby the existent comes more and more to embody those [real] generals which were just now said to be *destined,* which is what we strive to express in calling them *reasonable*” [CP, 5.433], [EP, 2:343]; see also [10, 64-65]. The highest ideal tentatively described by Peirce’s esthetics, then, is the quality of feeling evoked by the process that evolves greater reasonableness and harmony out of the plurality of things in the universe. In Peirce’s view, the highest ideal

[2] Jeffrey Barnouw traces this conception of esthetics to Friedrich Schiller, whose 1885 *On the Aesthetic Education of Mankind* (“The Aesthetic Letters”) Peirce read closely as a young man. Barnouw writes: “With his 1906 conception of esthetics Peirce in effect came back to the key idea he had discerned in Schiller’s *Aesthetic Letters*. . . an idea of ‘aesthetic determinability’ which few before or since have really grasped” [1, 161].

[3] I here use the word *nobility* to name the appropriate character of feelings. In its representation of what is true, good thought exhibits “veracity.” In its conformity to what is right, good action exhibits “propriety.” In its apprehension of what is admirable, good feeling exhibits “nobility.”

conceivable to us is not a state of absolute harmony or absence of strife – not nirvana – but rather *the feeling that accompanies* increasing order and harmony in the world of our experience.

2.2 Practics

The connection between esthetics and ethics is almost immediate. One can hardly embrace an idea of the highest good without attempting to direct one's actions toward realizing it, but the close connection between esthetics and practics is more than psychological. According to Peirce, it is above all a logical connection: "the instant that an esthetic ideal is proposed as an ultimate end of action, at that instant a categorical imperative pronounces for or against it" [CP, 5.133], [EP, 2:202]. This Peircean categorical imperative is not quite the same as that described in Kant' s famous formulae. In Peirce's view, the Kantian categorical imperative derives from Kant's particular conception of the *summum bonum.* While the Kingdom of Ends is a powerful and important conception of the categorical imperative, the principle of fallibilism indicates that neither it nor any other ought to be accepted as the last word on the matter.

Although Peirce admired Kant's ethics, he offered one serious objection to Kant's theory: the categorical imperative described there is not presented as being subject to criticism. "Kant, as you know, proposes to allow that categorical imperative to stand unchallenged - an eternal pronouncement. His position is in extreme disfavor now, and not without reason" [CP, 5.133], [EP, 2:202]. Peirce's normative science of practics is an ongoing inquiry aimed at determining the ends toward which one's will ought to be directed. It encompasses "the purely theoretical studies of the student of ethics who seeks to ascertain, as a matter of curiosity, what the *fitness* of an ideal consists in, and to deduce from such definition of fitness what conduct ought to be" [CP, 1.600]. Esthetics asks what is good; practics asks what aspect of the good is the proper end of human action.

Though we have from Peirce a provisional account of the *summum bonum,* no finite being can realistically direct its will toward *universal* increase of reasonableness. "Accordingly," Peirce writes, "the problem of ethics [practics] is to ascertain what end is possible" [CP, 5.134], [EP, 2:202]. I read this statement to mean *what end is possible for finite individuals to pursue.* Peirce emphasizes the limitations of our situation: "Here we are in this workaday world, little creatures, mere cells in a social organism itself a poor and little thing enough, and we must look to see what little and definite task circumstances have set before our little strength to do" [RLT, 121]. Practics recognizes human finitude: the individual is an ineffective agent *iftaken in isolation.* This fact indicates to Peirce that right action necessarily involves exerting individual effort in concert with the efforts of the extended community: "progress comes from every individual merging his individuality in sympathy with his neighbors" [CP, 6.294], [EP, 1:357]. Our part of the *summum bonum* is expressed in Peirce's interpretation of the Golden Rule: "Sacrifice your own perfection to the perfectionment of your neighbor" [CP, 6.288], [EP, 1:353].

Aside from his insistence on fallibilism, Peirce's conception of the categorical imperative does closely resemble Kant's. Peirce emphasizes a certain universalizability or sustainability as the hallmark of right action:

> it appears to me that any aim whatever which can be consistently pursued becomes, as soon as it is unfalteringly adopted, beyond all possible criticism, except the quite impertinent criticism of outsiders. An aim which cannot be adopted and consistently pursued is a bad aim. It cannot properly be called an ultimate aim at all. The only moral evil is not to have an ultimate aim. [CP, 5.133], [EP, 2:202].

The Peircean categorical imperative might be formulated as follows: **The aims one pursues ought above all to contribute, in the long run, to the increase of order, harmony, and connectedness within one's community and world of experience.** Any action that neglects this imperative is ultimately pernicious. Such thinking led, for example, to Peirce's virulent condemnation of Andrew Carnegie's *Gospel of Wealth* in "Evolutionary Love," where he offers a harsh criticism of what he calls the "Gospel of Greed" [CP, 6.294], [EP, 1:357].

2.3 Logic

Peirce's work on logic is far better known, and is much more extensive, than that devoted to esthetics and ethics. Here we need only concentrate on two aspects of Peirce's logic. First, we must articulate its nature and position as one of the normative sciences. Second, we must bring out the non-foundational realism Peirce associated with logic. I suggest that the same realism Peirce discerns in logic must also apply to his esthetics and practics.

As was seen above, logic is the study of the conditions under which thought can reasonably be considered to conform to the ideal or standard of truth. Truth is a species of the Right, which in its turn is a species of the Admirable. To embrace a conception of the *summum bonum* implies that one ought to endeavor habitually to feel attraction to that ideal, to develop nobility of feeling. Moreover, one ought to endeavor habitually to act with propriety, in a manner that promotes the *summum bonum.* The inquiry into the proper ends of such actions is practics. Logic is the third component in Peirce's program for understanding and realizing the *summum bonum.* If practical goodness consists in actions that contribute to realizing the highest good, then logical goodness likewise consists in thoughts contributing to this end in their own mode: "logical goodness is simply the excellence of argument – its negative, and more fundamental, goodness being its soundness and weight, its really having the force that it pretends to have and that force being great, while its quantitative goodness consists in the degree in which it advances our knowledge" [CP, 5.143], [EP, 2:205]. Logic aims to articulate the conditions for veracity, under which thinking can reasonably be considered to increase order, harmony, and connectedness in the world of thought.

Thought, according to Peirce, always occurs in the medium of triadic signs. The proper function of a **sign** is to represent accurately an **object** to an **interpretant** – three terms are always involved in a successful representation. The world of our experience is a world represented in thought. The world we inhabit can be seen, then, as an incredibly complex web of signs in constant and dynamic interpretive transaction. This activity, according to Peirce, is tending toward an ideal state of complete and accurate representation – an all inclusive sign-relation [NEM, 4:239-40]. Such a sign-relation cannot in fact ever be accomplished, because such a sign would need to generate an interpretant of itself. Thus the process must either stop without completion, or else it must continue forever toward completion. This ultimate and ideal unity is, however, the *telos* of thought. Such an ideal, it is said, is like the stars that we steer by but never actually reach. Logic describes the patterns of association and interpretation of signs that tend toward truth, where truth is the accurate and complete representation of Reality. Normative logic, the third normative science, is semeiotic, the theory of signs and sign-action [CP, 1.444], [SS, 80]. It is much broader than *formal logic,* which Peirce identified as the Mathematics of Logic, a branch of mathematics.

The most important aspect of logic is that it concerns regularities and laws in thought and experience. Esthetics concerns immediate feelings, while practics concerns immediate feelings and particular actions. Logic concerns the immediate and the particular, as well as the general – and the regularities and laws it discerns may be found in the worlds of feeling, action, and thought alike. The determination of such generals in fluid and chaotic experience is the key to establishing order, harmony, and connectedness in the world. Logical goodness or "excellence of argument" is a function of the degree to which thought advances toward truth, conceived as a unifying representation of reality.

This generalizing and synthesizing activity of thought, when it is exercised with deliberate control, is the means by which we most effectively promote the *summum bonum.* On the one hand, logic describes the *method* of deliberate inquiry into the ends of feeling (esthetics) and action (practics). On the other hand, logical goodness is *itself* a further development of esthetic goodness and practical goodness. The process of valid inference not only advances us toward apprehending a truth, it is also at the same time a right action (because it is an act of synthesis conforming to the Peircean categorical imperative) and an instance of the *summum bonum* (because a feeling of increasing reasonableness, a "sentiment of rationality," ordinarily accompanies the process). In typical fashion, Peirce presents three distinct but interdependent areas of inquiry, which taken together describe the scope of value theory.

3 Non-foundational Realism

Perhaps the most important feature of Peirce's value theory, and of his philosophy in general, is its insistence on a non-foundational realist epistemology. This position is best explained in the context of his logic and philosophy of science, but it must apply in all areas of inquiry. Peirce's "scholastic realism" maintains that there is a universe of reality, and that the aim of thought and inquiry is to develop adequate and accurate representations of this reality. If inquiry follows a sound method of investigation, its erroneous conclusions can eventually be exposed and corrected. At the heart of a sound method of inquiry is of course valid reasoning about the matter at hand. Logic is the inquiry that takes good reasoning as its object, and not merely as its method.

Peirce writes that "The real... is that which, sooner or later, information and reasoning would finally result in, and which is therefore independent of the vagaries of me and you" [W, 2:239], [EP, 1:52]. A report of a unique phenomenon – such as a stone falling upward on one occasion – is a report of an unreal phenomenon, or else it is an incomplete or inadequate report of the operation of an as-yet-elusive general law of physical reality. If there is some odd exception to the law of gravity at work on one occasion, this too is a real feature of the universe that would "sooner or later" be disclosed to us through further inquiry. To say that it could never *in principle* be discovered why such a phenomenon reportedly occurred is precisely to say that the phenomenon is unreal. "And so those two series of cognitions – the real and the unreal – consist of those which, at a time sufficiently future, the community will always continue to re-affirm; and of those which, under the same conditions, will ever after be denied" [W, 2:239], [EP, 1:52].

The "scholastic" component of Peirce's realism is this insistence that general laws governing the universe are the objects of genuine knowledge. These laws can be represented in cognizable sign-systems, and in Peirce's view the laws thus represented are nothing other than the universals whose reality was maintained, against the nominalists, by Duns Scotus. This conception of the real, and of scientific inquiry as the project of constructing adequate and accurate (i.e. True) representations of reality, applies to all science – including the three normative sciences.

Peirce had ample opportunity and cause to work out the implications of applying his scholastic realism to logic. His scholastic realism leads him to prefer logical realism over psychologism or logical formalism. Just as the validity of the law of gravity is a question of what *will* happen when two or more massive bodies come near one another, "the question of the validity of reasoning is the question of how frequently a conclusion of a certain sort *will* be true when premises of a certain sort are true; and this is a question of fact, of how things are, not of how we think" [W, 4:1, emphasis added].

The other notable feature of Peirce's epistemology is that it is non-foundational. This is not to say that knowledge builds upon a dubious base, but rather that the "indubitable foundations" of any inquiry are only contingently undoubted [W, 2:247-48], [EP, 1:61]. The inquirer starts with a doubt, which is the motive to inquiry. But the doubt is always enmeshed in a matrix of undoubted, practically indubitable belief – in terms of which the doubt is conceived. This matrix includes established common-sense knowledge, more esoteric knowledge established by previous inquiry, and a few items necessary "if reasoning is to go on at all: for example, all that is implied in the existence of doubt and belief, and of the passage from one to the other, of truth and falsehood, of reality, etc." [W, 4:1-2]. No belief is immune to criticism and even outright rejection], however. In fact, Peirce insists that the pragmatic assent to belief in the matrix of truths as it is understood by the inquirer at a given time must be balanced by the principle of fallibilism – the awareness that even though we can, and probably do, possess much true knowledge of reality, "we can never be absolutely certain of doing so in any special case" [W, 2:239], [EP, 1:52]. Roofers, rocket scientists, and everyone else are right to assume the truth of the law of gravity – as long as we realize the need to throw it overboard in the event that experience clearly shows it to be mistaken. Until that happens, though, we are fully justified in accepting and using it as a component in our "foundational" matrix of belief.

Peirce himself witnessed and contributed to the overthrow of practically indubitable beliefs in both logic and mathematics. He was also at the center of an intellectual circle in Cambridge, led by Chauncey Wright, that assimilated Darwin's evolutionary biology in the 1850's. These experiences with the structure of scientific revolutions convinced Peirce that foundations are always needed for inquiry, but that even the firmest foundation may conceal a dry rot. Peirce's realism thus remains "non-foundational" in the strict sense that there are no absolutely certain or essentially necessary components in our body of knowledge.

The meaning of Peirce's non-foundational scholastic realism for logic is clear: logic endeavors to describe the principles that govern reasoning when it successfully attains true conclusions, but we can never be certain that our present understanding of logical principles is complete or correct *even as we use those principles to conduct our inquiries*. The meaning of Peirce's non-foundational realism for esthetics and ethics is less clearly indicated in his writings. It is here, however, the Peirce offers some of his most interesting suggestions for value theory.

Taking up Peircean ethics first, we are led to ask what a non-foundational moral realism would entail. Likewise, we must ask what would be involved in practics, the theoretical inquiry into morality. Just as logical realism supposes that the principles that lead inference from true premises to a true conclusion are facts concerning the relations among propositions, so must moral realism suppose that the principles that lead action to conform to what is Right are facts concerning the relations among actions and aims. In other words, what makes actions right or wrong is something objective: the value of an action has to do with how well it actually conforms to a moral order that extends beyond the finite group's knowledge of morality.

Yet our knowledge of morality must also be non-foundational, on Peirce's account. When we inquire about ethics, it is because there is some doubt about our way of evaluating actions. Some principle of conventional morality appears problematic, or we come to suspect that our accepted understanding of what is right may be inadequate. Practics affords the means of critical examination of conceptions relevant to the problem area. Once this critical examination has begun, the logic of inquiry places everything in jeopardy. Though only those conceptions that actually do appear problematic during inquiry are to be challenged, there is no conception that is intrinsically immune to possible criticism. Practics is an inherently radical science.[4] In genuine inquiry, any hypothesis may be suggested as a possible resolution of the present doubt. Theoretical inquiry in ethics may suggest that some of our cherished principles are in fact pernicious, though it may take remarkable experiences to bring us even to consider the suggestion. More often, inquiry will suggest that we are in need of reinterpretation and development: the notion of rights, for example, may need to be restricted or extended, or our assumptions about what effects do in fact result from some action (such as telling certain kinds of joke in the workplace) may need to be reconsidered. Practics asks what is involved in making our actions conform to an ideal. Because the ideal is a general conception (as are all realities, in Peirce's view), it is susceptible to further determination through inquiry. The answers to the questions practics poses may, then, in any given case involve changes in our principles for evaluating actions or in our accepted conception of the ideal itself. The aim, as with any heuretic science, is to develop a better representation and understanding of how things are in the world.

The same must apply to Peirce's esthetics. That esthetics would be non-foundational is not hard to envision. After the breakthroughs of modern art, it seems reasonable to regard esthetic thought as the deliberate effort to experiment and challenge received

[4] "Ethics, then, even if not a positively dangerous study, as it sometimes proves, is as useless a science as can be conceived" [CP, 1.667].

conceptions of the Good and the Beautiful. The notion that Peirce's *realism* can be applied in this area, however, requires some elaboration. What does it mean to speak of realism in esthetic theory? Perhaps only this: the process of finding and developing order in the raw materials (physical and cultural) in our universe of experience, and the feeling that accompanies this process, really is the highest ideal of human life. It is Peirce's view that, given a particular situation in place and time with its particular resources and problems, the essential good of our existence lies in fashioning a harmonious balance between the rich potentiality of order and the already-ordered. The fine arts exemplify this process quite directly, but the same ideal – the *summum bonum* – motivates all our activities, from the most abstract mathematical reasoning to the most concrete efforts at subsistence farming.

The heuretic science of esthetics asks only one question: "What is the highest good?" The function of logical investigation of this question is to bring each individual, and each generation, to an understanding of the best answer that can be given to this central question. Esthetics, then, is a perpetual science of a single question. It is the work of practics and logic to determine what significance its answer has for our conduct and thought.

4 Reason, Sentiment, and Nobility of Experience

Peirce saw the importance of distinguishing two kinds of question in considering the role of interest or desire in structuring our thought. The first concerns what our most basic desires are, where we get them, and how they motivate us to act. These are questions of practical fact that admit of psychological or even biological answers. The second kind of question concerns what desires we ought to entertain, how we ought to acquire them, and the role that our various principles and beliefs ought to play in motivating our actions. These speculative or theoretical questions are addressed in the normative sciences. Peirce's value theory is especially interesting because he proposed answers to both sorts of questions, and ventured to suggest how the two distinct realms, the practical and theoretical, interrelate.[5]

The desires that actually do motivate our actions are deeply embedded habits, so deeply embedded that Peirce considers them all "instinctive" whether they are acquired or innate.[6] "It is the instincts, the sentiments, that make the substance of the soul. Cognition is only its surface" [RLT, 110]. We act on instinctive and sentimental desires because

[5] Peirce's most direct discussion of the relation between reason and instinct, between theory and practice, appears in "Philosophy and the Conduct of Life," the first of his 1898 "Cambridge Conferences" [RLT]. Peirce wrote this lecture after it was suggested that he speak on some "vitally important" topic. Since Peirce viewed philosophy as a theoretical science that did not necessarily address such areas of life, he laid out his view that it is "vitally important" that scientists separate their work from practical concerns. Peirce's lecture is unfortunately filled with sarcasm, irony and overstatement; at least it reveals how he *really* felt about the relation of theory to practice. John K. Sheriff provides a very helpful discussion of Peirce's philosophical sentimentalism [11, 83-89].

[6] Peirce's account of the role of experience in shaping beliefs and habits suggests that there are very few innate instincts.

they are instantly ready guides to action. What these desires are is a matter not entirely of chance, because they are determined by the previous experience of the race and the individual. From the individual's perspective, these antecedents may appear largely accidental. From the species' perspective, though, we find that *most* instinctive behaviors are basically admirable and beneficial: think of parents' inclination to care for their children or the neighbor's selflessness in helping in time of need. Such situations admit of a high degree of moral certitude. Consider also the kind of conceptual certitude we enjoy when we find, or establish, useful binary oppositions and hierarchical orders of priority within an unfamiliar or confused area of knowledge. On the basis of this general reliability of instinctive behavior, Peirce argues in favor of "philosophical sentimentalism" when it comes to ordinary practical action and thought [RLT, 111].

Nobody has a problem with the maternal instinct, of course. It is when our established instinctive approaches motivate inappropriate and destructive actions, or generate fruitless theories, that we look to philosophical reason for critique and reform. John Michael Krois observes that "The blind 'morality' of all true believers is more readily understandable in terms of 'sentiment' than argumentation. The objection might be raised that these are exceptional, extreme cases, that supporters of such causes identify with them to a degree that is untypical of ordinary ethical consciousness. But, and this is Peirce's point, they are not exceptions. They are typical – albeit deviant – examples of ethical attitudes" [5, 34]. Western philosophy has tended to suggest that once reflective and critical reason has identified a better way, our actions ought to immediately change. From there it is only a small step to the Enlightenment dream of a world in which all of our conduct, *all* our moral attitudes, arise from deliberate, reasoned decision: thus was born the economic fiction of the "rational consumer" who brings full information and ratiocination to every choice, for example.

Peirce was astute enough to realize three problems with this dream. First, it is not *necessary* to reason about every one of our actions. If the baby is hungry at 3 a.m., we simply get up and feed her. Second, it is not *possible* to reason about every action: time does not always permit it, and in many cases reason is inconclusive even though a decision must be made. Finally, Peirce points out that it is often not even *desirable* to do so: individual ratiocination is highly fallible in matters of "vital importance."[7] The twentieth century has suffered far too many ill effects of a misguided trust in experts who would reform the world on the basis of their theoretical expertise. Kenneth Laine Ketner proposes the following illustrations of this phenomenon, though there are countless others: "Consider the widespread destruction of our planet's environmental resources, or the use of rationalized systems of life (such as are reflected in political bureaucracies, collapsing Stalinism being a ubiquitous instance) in place of traditional systems of life" [4, 9]. Blind moral certitude can be destructive, but as Camus observed in *The Rebel,* there is an extra dimension of horror involved when the certitude is "justified" by philosophy, when reason is used to transform "murderers into judges." Philosophy and reason themselves are rendered impotent when they "justify" the specter of "slave camps under the flag of freedom" [2, 3-4]. Peirce goes so far as to say that, compared to

[7] This last point is especially important in response to Deweyan criticisms of Peirce's separation of theory and practice, such as that presented by John Stuhr [12].

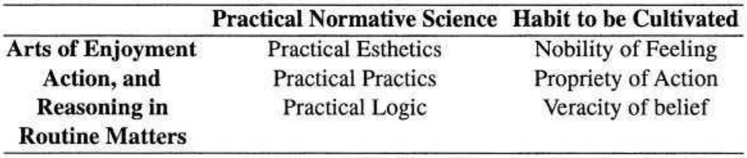
the errors of limited reason, instinct and sentiment are “practically infallible” guides to action in ordinary affairs [RLT, 111].

How, then, should reason, in the form of theoretical science, influence action? Peirce suggests that the channel of influence here is as slow and sure as the method of science itself:

> Instinct is capable of development and growth, – though by a movement which is slow in the proportion in which it is vital;... [I]t chiefly takes place through the instrumentality of cognition. The soul’s deeper parts can only be reached through its surface. In this way the eternal forms, that mathematics and philosophy and the other sciences make us acquainted with will by slow percolation gradually reach the very core of one’s being, and will come to influence our lives. [RLT, 121]

I suggest that we recognize two kinds of normative sciences: alongside the Heuretic (theoretically oriented) Normative Sciences described in Table 1, there are also Practical Normative Sciences, as indicated in Table 2. These relate in exactly the same way that the heuretic sciences of (theoretical) physics or mathematics relate to the practical science of engineering. Practical sciences use results from the heuretic sciences to attain some specific end other than the increase of knowledge about the world. These ends may be regarded as habits in the Peircean sense (think of predictably fluid traffic flow patterns or the stability of a bridge, which are the concrete instantiations of general laws in designed systems).

Table 2. The Influence of Reason on Habit



	Practical Normative Science	Habit to be Cultivated
Arts of Enjoyment Action, and Reasoning in Routine Matters	Practical Esthetics	Nobility of Feeling
	Practical Practics	Propriety of Action
	Practical Logic	Veracity of belief

In like manner the practical normative sciences use theoretical knowledge about esthetics, ethics, and logic to achieve particular ends – to cultivate habits of feeling, action and thought.

One example would be for us to use Peirce’s system of existential graphs to teach basic critical thinking skills to students, who then exhibit the habit of not committing common fallacies in their own thinking (see [3]). In acquiring familiarity with the rules and tools of sound thinking they are at the same time internalizing both ethical and esthetic lessons. There is an “direction of learning” to this process that reflects the interdependence of the practical normative sciences: learning the logical techniques of *how* to think well brings one to understand the ethical imperative that one *ought* to think well. Various successful and unsuccessful *efforts* to think well bring one, in turn, to appreciate the esthetic *experience* of what it is to think well or not. Thus the last stage in the direction of learning good reasoning is esthetic discernment. One comes to recognize the attractiveness or elegance that expert inquirers associate with certain ideas or theories; one's ear becomes attuned to their "ring of truth."

5 Systematic Abduction: Toward a Logic of Creativity and Discovery

Peirce's account of the role of the normative sciences in systematic inquiry suggests that there is a describable structure to the process of creative discovery. It is not guided merely by arbitrary and irrational accidents of history, but neither is it a narrowly logical process. Discovery is rooted in a thick "tacit knowledge." This includes finely attuned ethical and esthetic sensibilities alongside well-ordered technical knowledge. The veteran inquirer who has internalized the logical, ethical, and esthetic norms of a field is able to retrace the direction of learning, to follow it backwards. This process begins with the esthetic insight that an established order, a habitual way of structuring the world, is unsatisfactory in some way: an accepted theory does not "sit right," respected social conventions seem suddenly intolerable, trusted procedures of inquiry feel sterile or misguided. A creative response can emerge from such esthetic insights.

Here we confront the question of the basis of originality and creativity. Peirce maintained that all inquiry, the "struggle to attain a state of belief" or settled habits of thought and action, arises from a genuine experience of doubt [EP, 1:114]. Inquiry begins with the suggestion of a new hypothesis about how things are in the world. Peirce identified hypothesis formation, or abduction, as a distinct form of inference. He did not make much progress toward describing and analyzing the structure of abductive inference, and perhaps we cannot hope to fully understand the logical and psychological processes by which original and true ideas are developed. As a starting point I suggest that abduction is often initiated by a specific kind of doubt, namely, an awareness of esthetic dissonance. From this starting point the inquirer traces the direction of learning backwards through ethics to discover alternative logical and conceptual configurations. As the details of these alternative orders are worked out, two esthetic phenomena emerge. The first is the distinctive esthetic flavor that accompanies the alternative order. The second is the feeling that accompanies the development of increasing order and harmony, as the details of the alternative order are worked out. This phenomenon, I think, is the Peircean esthetic *summum bonum* itself.

Let me conclude this discussion of the esthetic ground of ordered thought with some speculation about how the process works, following the lead suggested by Peirce's account of the normative sciences. I will first consider an illustration of how this process may unfold in specific cases in the sphere of ethics and politics, then sketch a general model of the process. This may in turn suggest a way to more fully articulate the logic of abduction.

Consider the difference in experience that must exist between ordinary people and such social reformers as Thoreau, Tolstoy, King, and Gandhi. Most people are conventionally moral: they generally adhere to established norms of thought, conduct, and perception. Other ordinary people regularly violate those norms, perhaps out of a sense of rebellion or because of moral blindness, or for some other unreflected and ultimately indefensible reason. Though these may be censured as immoral persons, or punished as criminals, they too are functioning *within* the sphere of established norms: neither rejection of nor blindness to an established order constitutes fundamental change to that order. Social reformers likewise challenge established conventions, but they do so for admirable reasons: their challenge is directed toward realizing a novel and ethically defensible *telos.* From the esthetic insight that something is "out of order" they begin a systematic critique of established sensibilities. This leads to experimentation with actions that give rise to a different order of things, that follow a different logic of events, and that come with a different esthetic flavor. Thoreau's objection to a war led him to question the comfortable esthetic sensation of being a law-abiding citizen (he does seem to have been disposed to question this kind of comfort in any case), and to consider the alternative action of not paying his taxes on principle. This was Thoreau's creative move: his one act of non-payment established the novel general idea of civil disobedience as a citizen's duty. Once created or discovered through a process of experimentation that begins in mere possibility, civil disobedience entered into the established reality of the social and political order. It became a recognized way to advance the *telos* of the society in certain circumstances, and it has its own esthetic flavor that can be recognized and experienced by others. Once a part of the established order, civil disobedience itself becomes a matter of interest and critical analysis. In their efforts to understand it as a political strategy, people come to recognize the ideals promoted by civil disobedience (such as freedom of individual conscience, and the fundamental democratic right to non-revolutionary opposition to government) as legitimate ideals in themselves. These newly-prominent ideals define a new institutional order in the society. A similar speculative story might be told for other significant movements toward social reform, such as Tolstoy's esthetic anarchism or Gandhi and King's development of nonviolent direct action as a response to institutional violence and oppression.

We may discern a common pattern in these movements toward ethical and social reform. The abductive inference originates in an esthetic insight about the wrongness of things and leads to a deliberate and specific change in conduct. The experimental change in conduct opens the way to an alternative conception of ordered social reality. Once formed, this alternative conception is recognized as esthetically more noble than what it replaced. The struggle to bring it into being, too, is characterized with the esthetic quality peculiar to the *summum bonum.* This esthetic apprehension of the general movement toward greater harmony and reasonableness is a crucial aspect of what drives the reform movement forward. On the other hand, when this esthetic flavor is lost, the individuals who were promoting the reform lose hope and energy. The movement may die of a kind of esthetic starvation.

These speculations about the basis of social reform have probably already placed us too far out on a limb of inference for our own good. Finding ourselves this far out, though, we may as well inch along to the end. I suggest that the pattern of creative discovery just described may be typical of all fruitful abductive inference, whether the sphere of inquiry is moral and political, or artistic, or scientific. Constructive innovation and true discovery are by no means inevitable aspects of change: history shows that we can regress as well as progress. History also shows that happy accidents play a major role in much of what we consider progress. On the view I propose here, though, progress toward the true and the good is not *merely* the accidental result of irrational power relations and historical contingencies. They occur when competent inquirers systematically critique and then investigate the implications of anomalous esthetic insights. Such critique and investigation serves to establish defensible new interests that influence action; these actions allow us to develop the concrete reality of a new order of things, which embodies a fresh esthetic sensibility.

Peirce's theory of the normative sciences is a framework of relations among esthetic insight, practical action, and logical structures. The non-foundational realism of his theory suggests two crucial things about creative activity. First, it may be possible to articulate a common method for fruitful abductive inferences. We may be able to describe a logic of discovery grounded in esthetic insight. Second, the Peircean framework allows us to critically evaluate proposed innovations to established systems of order in terms of their motivating impulse and emergent *telos*. This principle should apply whether we are considering "practical" systems like social and political institutions, or "logical" systems like philosophical approaches and scientific theories. Although some changes are merely expressions of power or the results of historical contingencies, others are the fulfillment of a new order of things whose value can be publicly understood and critically judged even as they are unfolding.

References

[CP] Peirce, Charles S. *Collected Papers of Charles Sanders Peirce.* 8 vols. Vols. 1-6 ed. Charles Hartshorne and Paul Weiss. Vols. 7-8 ed. Arthur Burks. Cambridge: Harvard University Press, 1931-58.

[EP] Peirce, Charles S. *The Essential Peirce.* 2 vols. Vol. 1 (1867-1893) ed. Nathan Houser and Christian Kloesel. Vol. 2 (1893-1913) ed. The Peirce Edition Project. Bloomington, Indiana: Indiana University Press, 1992, 1998.

[NEM] Peirce, Charles S. *The New Elements of Mathematics.* 4 vols. Ed. Carolyn Eisele. The Hague: Mouton Publishers, 1976.

[RLT] Peirce, Charles S. *Reasoning and the Logic of Things: The Cambridge Conferences Lectures of 1898.* Ed. K. L. Ketner and H. Putnam. Cambridge: Harvard University Press, 1992.

[SS] Peirce, Charles S. and Victoria Lady Welby. *Semiotic and Significs: The Correspondence between Charles S. Peirce and Victoria Lady Welby.* Ed. Charles S. Hardwick. Bloomington: Indiana University Press, 1977.

[W] Peirce, Charles S. *Writings Of Charles S. Peirce: A Chronological Edition.* 5 vols. to date. Ed. Edward Moore, Christian J. W. Kloesel, et al. Bloomington: Indiana University Press, 1982–.

1. Barnouw, Jeffrey. "The Place of Peirce's 'Esthetic' in his Thought and in the Tradition of Aesthetics." In [8, 155-178].
2. Camus, Albert. *The Rebel.* Trans. Anthony Bower. New York: Vintage International, 1984.
3. Forbes, Morgan. "Peirce's Existential Graphs: A Practical Alternative to Truth Tables for Critical Thinkers." *Teaching Philosophy* 20 (December 1997): 387-400.
4. Ketner, Kenneth Laine. "Peirce's Sentimental Conservatism." In *Logic and Political Culture.* Ed. E. M. Barth and E. C. W. Krabbe. New York: North-Holland, 1992. 3-9.
5. Krois, John Michael. "C. S. Peirce and Philosophical Ethics." In [8, 27-37].
6. Kuhn, Thomas S. *The Structure of Scientific Revolutions.* 3rd edition. Chicago: University of Chicago Press, 1996.

7. Parker, Kelly. “Charles S. Peirce on Esthetics and Ethics: A Bibliography.” 1999. http://agora.phi.gvsu.edu/kap/CSP_Bibliography/CSP_norm_bib.pdf
8. Parret, Herman (ed.). *Peirce and Value Theory: On Peircean Esthetics and Ethics. Semiotic Crossroads.* Vol. 6. Philadelphia: John Benjamins Publishing Company, 1994.
9. Popper, Karl R. *The Logic of Scientific Discovery.* 5th ed. New York: Routledge, 2002.
10. Potter, Vincent G., S. J. *Charles S. Peirce on Norms and Ideals.* Amherst: The University of Massachusetts Press, 1967.
11. Sheriff, John K. *Charles Peirce’s Guess at the Riddle.* Bloomington: Indiana University Press, 1994.
12. Stuhr, John J. “Rendering the World More Reasonable: The Practical Significance of Peirce’s Normative Science.” In [8, 3-15].

Memes as Signs in the Dynamic Logic of Semiosis: Beyond Molecular Science and Computation Theory

Terrence W. Deacon

Department of Anthropology and Neurosciences Program
University of California, Berkeley

Abstract. The concept of meme misidentifies units of cultural information as active agents, which is the same "shorthand" that misleads our understanding of genes and obscures the dynamic logic of evolution. But the meme concept does offer hope by contributing something missing from many semiotic theories. In treating memes as replicators, Dawkins fails to distinguish mere patterns from information (and replication from interpretation), which leads to the problem encountered in all realms of information processing: what counts as information is context dependent. Nothing is intrinsically meaningful, to be so it must be interpreted. In the evolution of both genes and words, replication has always been a multilevel affair in a dynamic system, from which what we conceive as "adapted" or "interpreted" units emerge. Memes are replicas not replicators, and I suggest that the iconic function of signs, as identified by Peirce, is the essence of the meme concept. As in sign function, both gene and meme functions are informed by whatever relationship exists between the physical pattern of the token and the system of processes in which each is embedded (so these are semiotic relationships). I argue that two, not-clearly-articulated aspects of the meme concept could rescue semiotic from mere descriptive taxonomy and lead the way for a general theory of semiosis and a unifying methodology for the semiotic sciences to emerge.

1 Introduction: The Logic of Replicators

Few scientific terms introduced into scientific and popular vernacular have enjoyed the impact on intellectual debate as has the term "meme." The meme concept, introduced by Richard Dawkins in his influential book The Selfish Gene [1], has spawned a vast following of "memetics" enthusiasts and not a few books expanding on Dawkins' original insight at great length. This popularity owes much of its widespread appeal to Dawkins' corollary characterization of genes as "replicators" and his reformulation of evolutionary theory from a "gene's-eye" perspective. His analysis of replicator selection in biology led directly to his proposal that one could also think of social-communicative processes as though there were a gene-like unit of replicator selection driving the evolution of cultures: the meme.

The meteoric rise of molecular biology to preeminent status as the epitome of cutting edge scientific-technical progress of the last half of the 20th century has only been matched by that of the information sciences. The serendipitously parallel explosion of knowledge in these two fields has provided the perfect context for Dawkins' reframing of evolution in molecular-informational terms. Dawkins, following the lead of such

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 17–30, 2004.

pathbreaking biological theorists as W. D. Hamilton, G. C. Williams, and E. O. Wilson, shifted the focus of evolutionary thinking to genes themselves as the ultimate unit of biological information and as the locus of natural selection, rather than traits (phenotypes), whole organisms, or populations, as the then current evolutionary theories assumed. Organisms were characterized by Dawkins as "lumbering robots" designed by the genes to carry out their (the genes') reproductive imperatives, because after reproducing most organisms perish. Even offspring in sexually reproducing species are not copies. In contrast, genes, or more precisely the sequences of information embodied by them, are conserved from generation to generation. They are passed down minimally modified, if at all, across countless generations. In this way gene sequences retain the possibility of near immortality. Evolution thus appears to be about preservation of gene information at the expense of all else.

According to Dawkins, the critical defining feature of a gene that makes evolution possible is its ability to make precise copies of itself. Focusing on this remarkable molecular predisposition, Dawkins described genes as "replicators": units of information capable of copying themselves and also capable of influencing the probability of being copied in other indirect ways. He argued that this propensity is the most fundamental factor in biological evolution. Evolution could thus be reduced to a kind of molecular information processing based upon this one basic operation - molecular biology meets computation theory.

The patterns exhibited in evolution, then, could be described as merely playing out the statistics of replicators competing with one another. Replicators that, besides merely copying themselves, also happened to exert some additional positive influence on the probability of their own replication, thus biasing the statistics, would be favored over those with less positive effect or none at all; hence genes could be said to exhibit a kind of "selfishness." There are many ways that such biasing can occur. Virus genes produce structures that fool other organisms into treating the virus genes as their own and copying them willy-nilly; genes in multicellular bodies produce structures that increase the probability that the organism will reproduce and thus be likely pass on copies of these genes; and sometimes these organism behaviors may even increase the probability that kin will reproduce, and pass on copies of both gene sequences inherited from parents. The "genes-eye-view" of evolution, then, was supported by the recognition that in all these cases, making copies of genetic information, not organism structure, was paramount. Everything else tended to be sacrificed before gene fidelity, fecundity, and longevity in evolution.

Framing this core feature of life in such general terms, also led Dawkins to consider (almost as an afterthought) that genes might not be the only potential replicators in the world. Cultural information capable of being passed on to others by mimicry or copying could also qualify. He named this sort of replicator a mimeme (unit of mimesis) or meme for short. Any cultural artifact or habit that could be transmitted by being copied could qualify as a meme. Dawkins offers such examples as popular songs, fads, mannerisms, recipes, tools, etc., and other memeticists have even suggested that languages, religious traditions, foraging techniques, and folk stories (to name just a few of thousands of suggested examples) might be considered memes. By Dawkins' own reckoning over two decades later the term "meme" has indeed exemplified itself in being propagated at an unusual rate for a newly coined jargon term [2: xiv]. In his search of mentionings of the term on the world-wide web, he noted half a million mentions as of August 1998, out-reproducing by orders of magnitude such recently coined jargon words as "sociobiology" and "spin-doctor."

2 Memetic Difficulties

But memes have not won wide acceptance as a useful theoretical construct in most fields where they might be relevant. Instead, with a few exceptions they remain a sort of analogical stand-in for concepts not fully fleshed out and so unsuitable for precise analysis. In a recent overview, Dawkins and Blackmore offer three commonly cited areas of disagreement and unresolved definitional problems regarding memes [2: x-xvi, 53-66].

(1) Memes appear to have insufficient copying fidelity to be evolvable. This problem long troubled geneticists prior to the discovery of the molecular basis of genetic information, because if the units of inherited information could mix or grade into one another evolution would quickly regress toward the mean and grind to a halt. Information would be rapidly eliminated and subtle deviations never accumulated.

(2) Nobody really knows what a meme physically is. Dawkins and others talk about memes as information. And in many contexts writers often substitute the term "idea" for meme, since memes are defined as whatever is passed on when something is copied by one person from another, either directly or indirectly. This is evident when writers describe memes with such phrases as "ideas begin to 'have a life of their own'" [2:29] and "a unit of information residing in a brain" [3: 47]. And is also made evident by the plethora of near synonymous terms (e.g., cultural traits, culturgens, corpuscles of culture) and the disagreements as to whether only copied information qualifies or whether any transmissible cultural feature qualifies.

(3) There is difficulty finding agreement on how large a unit deserves the name "meme." Is it as small as a word or a morpheme (root meaningful unit constituting a word) or a type of handle on a pot, or as large as a language and a religious tradition?

(4) Finally, some have questioned whether there is a memetic parallel to the genotype/phenotype distinction in biology. For example, is a recipe the meme and the cake the memetic phenotype; is a performance of a song the phenotype of the remembered idea of the song?

Both Dawkins and Blackmore summarize and address these problems in a recent discussion [see 2]. They conclude that all are in one way or another unresolved, but that they are either non-problems and can be ignored or else can be shelved for the time being, awaiting some memetic Watson and Crick to sort out their material basis.

I am far from convinced by these responses, but won't attempt to recount them or systematically critique them here, and argue only that we cannot shelve these problems and still have a science of memetics. To do so abandons this concept to remain the subject of pop science and internet discussion groups, but not of technical research. Though some reviewers would recommend this, I think there is a baby somewhere in this cloudy bathwater. To rescue it I wish to focus on a couple of tacit assumptions that

I think underlie these persistent problems and to suggest a change of perspective that may help clear up what exactly we are talking about.

3 Core of the Problem

The trouble with memes, as I see it, is that they have been misidentified. I am convinced that there is indeed something that bears social-cultural information and which evolves by means that are roughly analogous to processes of natural selection acting on DNA sequences and the living processes linked to them. That something is not, however, ideas. It is not something residing in brains. And it is not something imbued with the power to affect its own replication (but then again, neither is a naked strand of DNA, except in very special laboratory processes called PCR). It is not some new class of entity hitherto unrecognized, unstudied, and unnamed.

To state my conclusion at the outset: Memes are signs, or more accurately, sign vehicles (or representamens, to use Charles Peirce's clumsy but precise terminology). They are not some intangible essence that is passed from brain to brain when something is copied. Signs are concrete things, or events, or processes and they have been categorized and studied systematically for centuries. Memes are thus the subject of a body of theory that Peirce called Semiotic!

Having said this, one might think that I will conclude that memetics is much ado about something already steeped in a vast literature and a rich theoretical basis, and to which memes have little to offer but an alternative terminology. But I don't think so. The meme concept has generated recent excitement precisely because it seems to offer hope of providing something that other theories of social and semiotic processes have not succeeded in providing. It address the process of semiosis, i.e., the dynamical logic of how the symbolic and concrete constituents of culture arise, assume the forms they assume, and evolve and change over time. This ambitious promise can only be met, however, if the problems with memetic theory are remedied and if a synthesis within a broader semiotic theory can be realized [see Note 1]. This is a tall order.

So, beyond recognizing this correspondence, what needs to be done? First, we must use this insight to remedy some of the theoretical problems that plague the meme concept, and second we must use it to discover what is uniquely emphasized by the meme concept that can contribute to a theory of semiosis and which may be overlooked or misunderstood in current semiotic analyses.

The core problem of this theory, I think, is a kind of misplaced agency, that gives the impression that both genes and memes - replicators - can be understood without considering their embeddedness in a dynamic system which imbues them with their function and informational content. This, then, is not just a problem with memes, but a problem with the replicator concept in general, inherited from Dawkins' short-circuited description of information processes in biology. I say "short-circuited" because it is not wrong, it just cuts corners that suggest that certain essential aspects of information processing in biological systems can be treated as merely derivative from the replicator concept. In fact, this inverts the reality.

Though an anthropomorphic shorthand has often been used to describe replicator "behavior" (e.g., it is common to read such phrases as "for the benefit of the gene, not the organism" or "genes promote their own self-interest" or "genes act selfishly," etc.), this phraseology was never presumed to project any purposeful or even animate character on genes. Genetic replicators are just strings of DNA sequence information that happen to get copied and passed on to the future successfully. Genes are not active automatons, they are not "agents" of evolution; but are just structures, passive information carriers that can become incorporated into biochemical reactions, or not, depending on the context. Referring to genes as though they were active agents promoting their own copying process is a shorthand for a more precise systemic account of how a gene's effect on its molecular, cellular, organismic, social, and ecological context influences the probability of the production of further copies. Taken out of this almost incomprehensibly rich context - placed in a test tube, for example - DNA molecules just sit there in gooey clumps, uselessly tangled up like vast submicroscopic wads of string.

So what is wrong with this shorthand, if everyone agrees that genes are just patterns of DNA sequence information? It is the information content that defines a replicator, not its material or even energetic embodiment. The physical gene (i.e., the string of atoms comprising the DNA molecule) is not what evolution conserves and passes on. It is the information, embodied by the nucleotide sequence that is the point. Well, the problem is that the DNA sequence is not intrinsically information, it is just a pattern. Indeed, much of the sequence information in my genes may never be treated as information in the construction and operation of my cells, even if it may at some point become information for evolution. What makes a pattern information? And how does some information become involved in its own replication?

4 Replication in Context

The problem of defining the difference between mere pattern and information is encountered in all realms of information processing and has the same unhelpful answer: what counts as information is context-dependent. The information some pattern can convey is both a function of its distinguishable physical characteristics with respect to other patterns and of its embeddedness in a living/semiotic system that can incorporate this into its functional organization. No thing is intrinsically meaningful, it is interpreted to be so. And interpretation is more than replication - yet these two processes are intimately related.

We are all familiar with the fact that in different interpretive contexts the same typographical words or spoken phrases can function quite differently. This can be true of homophones used in different sentences or of cognates in different languages that have historically diverged in their denotations and connotations while their phonological or typographical forms have remained the same. In genetics this is also evident. Gene sequences (homeobox genes, for example) that contribute to the embryogenetic assembly of heads and brains in flies have almost identical counterparts in humans, so much so that inserting the human gene into mutated flies with the corresponding gene damaged can partially restore normal fly head development. But notice, that it does not produce human heads (as in a science fiction parallel). The cross-species sequence similarity (and evolutionary common ancestry) is essential to this function, but the information is interpreted in a fly context.

In both the evolution of words and of genes, these units of pattern have been preserved over time because of the ways they contributed to the usefulness and replication of the larger systems in which they were embedded. Replication has always been a multilevel affair. It is not a function vested in a gene or in a word, or even in a complex of genes or words, but in a dynamic system in which these are only one kind of unit. It is only in certain unusual and degenerate cases, such as viruses or their computer counterparts, that we can provisionally speak of the functional interpretation of the pattern being self-replication itself, and then only because we recognize that even this function is parasitic and depends on a short-circuiting of a more complex system's functional-interpretational organization. Viruses contain only self-referential replicative information, only if considered without their host.

The selfish gene perspective views genes on the analogy of viral genes, ignoring their implicit parasitism. The caricature of genes as agents competing to inhabit organisms (and the parallel memetic version: "The haven all memes depend on reaching is the human mind" [4: 207]) brackets out any consideration of the systemic origin of gene (and meme) information, as well as its means of replication. This tacitly suggests that the system in which a replicator is embedded can be treated as a passive vessel and the information that constitutes the replicated unit can be understood independent of this context. It assumes that gene/meme replicators are primary and primordial and that the systems in which they are found are derived solely from their interactions.

Quite the opposite is true in both biologic and semiotic processes. These units actually emerge out of the systemic relationships, crystallize out, so to speak, as regularities and bottlenecks in the passage of information become entrenched in the course of these systems' evolution [see 5; Note 2].

Life did not begin with genes. Most biologists now believe that DNA probably came into the process of evolution long after life began, as a more stable, more easily copiable, and functionally compartmentalized molecular storage medium, with proteins and RNA preceding. The sub-sequences along a DNA molecule that subsequently came to be recruited as "genes" are in this sense artifacts that reflect prior regularities of higher-order systemic molecular relationships comprised by different kinds of molecules than DNA. True, once life adopted this sequestered form of molecular information storage it changed the nature of the evolutionary process, but one should not make the mistake of thinking that this makes genes more primary than the functional chunking of the molecular systems to which they evolved to encode.

The same can be seen to occur with word change. Eliminative and deconstructive processes that simplify and subdivide what count as meaningful linguistic units, and synthetic processes that lump formerly separate meaningful units into larger unanalyzed wholes, are both at work in language change, driven by the social-discursive-typographical-pragmatic chunking that determines their selective advantage with respect to other alternatives (I won't speculate on whether there could have been a protolanguage before words, etc., that they replaced).

Genes and memes are not the locus of the replication process, nor are they somehow the functional unit of information. They are replicas not replicators. They are rather more like the concretion of information bottlenecks in a system. Like a computer operation that mechanically manipulates patterned elements irrespective of their interpretation as symbols and yet still produces outcomes that correspond to symbolic operations, the advantage of replicas that capture the functional chunking of a larger system is that they too can be operated on (e.g., mutated, recombined, and selected) irrespective of function and yet have functional consequences when interpreted into a system [see Note 3].

Recognizing this computational parallel leads to an important redefinition of what constitutes a meme. Once the meme concept is divested of virtual agency we find that it is not an idea, nor some "unit of information residing in a brain" [see 4], nor "something however intangible, ... passed on ... when we copy each other" [3: 52], it is not even the information encoding some cultural function, it is merely a physical pattern. It is something concrete, mostly outside of brains (though something like the instructions for making it may be in a brain), that can be easily and unambiguously copied, precisely because it is some tangible thing [see Note 4].

Genes and memes are the physical loci where the replicative and adaptational functions intersect, but these loci do not "contain" the information that constitutes these two functions any more than they determine their own replication. The information for both functions is constituted by whatever relationship exists between the physical pattern of the token and the system of processes in which each is embedded. This helps to resolve a number of the problems determining what constitutes a meme. It is not insubstantial. It is some artifact form or type. Its information content is not intrinsic. It is a physical pattern that can be both copied and mapped into the functional relationships of other systems. It conveys information via replication of this pattern directly or by having the pattern re-expressed (i.e., transcribed) in terms of features of some other higher-order system. Its physical boundaries are also not intrinsic, but rather are a function of the interpretive processes into which it gets recruited (as is the case for DNA sequence information, by the way). Bounded interpretation may nevertheless be aided if the artifact in question is itself physically bounded. Because this medium has its own physical and copying characteristics, irrespective of its information content, there is an effect of these physical constraints on the ways that the information it bears can evolve.

5 A New Synthesis: Replication in Semiosis

That returns us to the equation that started this critical evaluation. A meme is a sign: some physical thing which, by virtue of some distinctive feature, can be recruited by an interpretive process within a larger system as re-presenting something else, conveying information into that system and reorganizing it with respect to that something else.

So what does the meme concept add to semiotic analyses? The answer is that it could rescue semiotic from being a merely descriptive taxonomic enterprise by leading the way to a theory of semiotic causality and of the generation of representational forms and interpretive systems. The key to this derives from two prominent aspects of meme theory that are not so clearly articulated by most semiotic analyses: (1) a focus on the primacy of replication, or literally "re-presentation" for grounding (evolving) referential processes (functions); and (2) the recognition that re-presentational relationships emerge from competitive/cooperative evolutionary processes which determine the form and persistence of the interpretive systems in which they are embedded. Though in this short essay I cannot hope even to outline such a theory [see Note 5], I will conclude with a few suggestive comments on the direction such an enterprise might take with respect to these overlapping conceptual issues.

First, let's consider the primacy of replication in semiosis. According to Peirce, "The only way of directly communicating an idea is by means of an icon; and every indirect method of communicating an idea must depend for its establishment upon the use of an icon" [6: 2.278]. Another aspect of this same relationship can be grasped by noticing that when you communicate an idea to someone else, the fact of its being communicated and understood consists in its being replicated, or re-presented in the other so that there is now a kind of implicit iconism between what is in your mind and in their mind. This is also the essence of the meme concept.

Peirce intends to convey more than this, however, by suggesting that communicating via icons or by indirect reference to icons is how all communication is inevitably grounded. This is because all other forms of reference are indirect and dependent on reference to icons. This hierarchic relationship is critical, and is not fully spelled out in meme theory nor is it agreed upon by semioticians. I give a superficial account of it in my book The Symbolic Species [see 7: 69-101]. To summarize three chapters of that book in a single sentence: indices are constituted by relationships among icons (and are thus indirect re-presentations) and symbols are constituted by relationships among indices (and are thus doubly indirect re-presentations). In other words, to interpret any sign is ultimately to analyze it to its constituent iconic basis. So at this level of analysis, mimesis can be seen as both the primary mode of information transmission and also the ground of interpretation.

In evolutionary terms, these two relationships - replication of the significate pattern and transcription of it into some other medium and with respect to some systemic adaptation - also overlap when it comes to the explanation of evolved forms. Natural selection favors the replication of genetically encoded information that in some way or other internalizes critical features of the organism's environmental context that affects its replication (e.g., up-regulation of microbial enzyme production in response to the increased concentration of a food substrate). Thus an adaptation is a kind of mapping of internal organization to some relevant environmental feature: a representation of the outside inside. The "goodness" of the representation of this external feature is thus pragmatically assessed with respect to its effects on the probability of replication of the DNA sequence patterns that helped generate it. In this way adaptation is defined with respect to gene replication and vice versa, producing linked replications at two levels, but neither makes sense without the other. Without any (even indirect) adaptation genes have no information; without some substrate to embody and replicate the information implicit in the adaptation there is nothing for it to be an adaptation for.

A parallel argument can be made for semiosis in general. The ultimate significance of a sign is grounded in the consequences of the system of interpretive habits and dispositions it generates and is generated by (i.e., in the larger context of patterns of sign usage of which it is a part). As in biology, then, the relative prevalence and probability of a sign's use will be a function of the correspondences between its significant consequences and some relevant aspects of the larger context in which it is embedded. Peirce himself appeared to recognize this in his later writings, hinting that pragmatism might naturally follow from a universal evolutionary logic.

Until now, classic semiotic theories have not had much to say about why certain signs persist and others do not, or why certain semiotic systems evolved the forms they now exhibit. Indeed they have been mostly synchronic and descriptive in their intent. But if the evolutionary analogy is appropriate here, then we may be able to understand fully the structure of semiotic systems and the types of sign relationships and interpretive processes that constitute them only if we analyze them as products of competing replicative processes: signs competing with signs. However, rather than competing for space in human memory, signs must be understood as physical markers competing for representation in the physical media of the semiotic systems that surround us. Their incorporation into brains is not the measure of signs (memes) evolutionary "success," as though signs were parasites. Rather, like the interpretation of genetic sequence information when it is transcribed into the machinery of the cell, a sign's replication depends upon its effect on what brains do, via the behavioral adaptations the signs promote. To the extent that this behavioral or cognitive phenotype increases the probability that the signs will be produced and used again, especially by others emulating similar functions that also make use of them, they will be spread and stabilized in the semiotic corpus.

Semeotic theories since Peirce have distinguished between the replica and the functional aspects of sign relationships. Specifically, Peirce recognized that one could restrict analysis to consider only the physical qualities that serve as sign vehicles without explicit consideration of their relationships to their referent objects or interpretation (an enterprise Charles Morris called Syntactics). Computation theory and mathematical information theory make use of this analytic compartmentalization. And it is to some extent what meme theory attempts as well. However, in these cases it is tacitly assumed that the items under study are already involved in interpretive processes and bear significance to a larger system. Provisionally "forgetting" this antecedent assumption (as is the case here) has often led theorists in all three domains to claim primacy of this lowest level of analysis. So, for example, the idea that cognition is nothing but "computation" - the rule-governed mechanical manipulation of markers according to their shapes - is a parallel fallacy to the memetic fallacy [see Note 6].

Peirce also argued that one could analyze referential relationships without explicit consideration of their relationships to the pragmatic consequences that are their ultimate basis of interpretation (a realm Morris identified with Semantics). Though this hierarchic analytic strategy is implicit in the biological analogy from which the meme concept is drawn, it was obscured by the apparent autonomy of the replicator concept. But the systemic embeddedness cannot be totally ignored, even if a coherent analysis of only the "syntactic" aspect is possible. Both living processes and other semiotic processes imbue the patterns of their constituents (genes or signs) with informative power by virtue of their roles in a larger functional system. Something is only a sign (a meme, a gene) when interpreted, and interpretation is ultimately a physical process involving the production of additional significant patterns: replicating more signs (or "interpretants" as Peirce called them). In other words, sign interpretation is ultimately mediated by sign production (i.e., replication), as gene function (i.e., interpretation into an adaptation) is ultimately assessed with respect to gene replication. As the biological parallel suggests, this is basis for an evolution-like process that can lead to ever more complex systems of interpretation (adaptation).

6 Some Concluding Remarks

The power of recognizing that memes are signs is that it offers a bridge to help us recognize that biological evolution and life in general are semiotic processes, and conversely that semiotic processes are intrinsically dynamic physical evolutionary processes. Signs evolve, and they have pragmatic consequences, by virtue of which they are selectively favored to remain in circulation or become eliminated over time. It is by virtue of the memetic analogy to genetic evolution that we may discover the dynamical logic still required for a complete theory of semiosis, rather than merely a semiotic taxonomy.

Biology is a science concerned with function, information, adaptation, representation, and self-other relationships - semiotic relationships, not merely physical relationships. Biologists tend to focus on the physical relationships among the information-bearing units in living processes whereas semioticians tend to focus on the significate relationships of the units of human communication, but ultimately these apparently unrelated analyses in different domains are entirely complementary. Taking this correspondence between biological information processes and social information processes more seriously may provide some new insights into the dynamical logic by which representational relationships form and evolve. I believe that this new synthesis will be the crucible from which a general theory of semiosis will eventually emerge. The theory of memetics is not the answer to a theory of social and psychological evolution, but reinterpreted it may suggest some bridging concepts that can lead to a unifying methodology for the semiotic sciences.

Acknowledgment. I thank Mary Keeler for editorial assistance in preparing the manuscript for this paper.

References

1. Dawkins, R. [1989]. The Selfish Gene. Oxford U. Press.
2. Dawkins, R. [1999]. Blackmore Susan (Ed.). The Meme Machine. Oxford U. Press.
3. Dawkins, R. [1982]. The Extended Phenotype. Freeman.
4. Dennett, D. [1991]. Consciousness Explained. Little Brown.
5. Deacon, T. [2003]. "Multilevel selection in a complex adaptive system: the problem of language origins." In B. Weber & D. Depew (eds.), Evolution and Learning: The Baldwin Effect Reconsidered. MIT Press.
6. Peirce.C. S. [1931-53]. Collected Papers of Charles Sanders Peirce (8 vols.), Arthur W. Burks, Charles Hartshorne, and Paul Weiss (eds.). Harvard University Press.
7. Deacon, T. [1997]. The Symbolic Species: The Coevolution of Language and the Brain. W. W. Norton, and Co.
8. Deacon, T.W. [2003]. "The Hierarchical Logic of Emergence: Untangling the interdependence of evolution and self-organization." In: B. Weber and D. Depew (eds.), Evolution and Learning: The Baldwin Effect Reconsidered. MIT Press.

Notes

1. Such a theory would conceive language as an evolved biological phenomenon, with its core architectonic logic conceived in evolutionary terms; then our con-

ception of the relevant evolutionary processes may need to be considerably broadened to deal with the complexities involved.

The key concept for understanding this logic is what I will call “masking.” I am using this term to refer to shielding or protecting something from natural selection. Selection is often viewed as a negative or subtractive factor in evolution. This is explicit, for example, in Baldwin’s theory of evolution, and in part motivates his theory as a means to introduce a “positive factor” (Baldwin, 1896). But this is an oversimplification. Selection is generated by any bias affecting reproduction of genes or traits, positively or negatively. What links all these multilevel co-evolutionary theories together is that they involve differential exposure to selection: either as something is shielded from natural selection or else is unshielded and newly exposed to selection by virtue of some action of the organism itself.

By unpacking the causal logic of the competing evolution theories and recognizing the central role played by masking functions, and the way these distribute, collect, and reorganize selection factors, we have begun to glimpse the generality of these processes, and how they may actually contribute to such complex multilevel adaptations as language behavior. This can liberate us from the tendency to think in terms of single evolutionary causes and simple accounts of adaptive structures. It also creates a biological context within which to begin to consider the contributory roles of systemic self-organizing processes, without succumbing to the tendency to attribute Lamarckian functions to them, or in any way envisioning that they undermine the constraints and logic of natural selection. Natural selection is simply far more complex than is often appreciated. But it would be a mistake to imagine that we could reduce the logic of these processes to nothing but natural selection. The very topology of the process is fundamentally altered in this case. The simple feed-forward of phenotypic effects on reproductive success has become embedded in another feed-forward loop in which other features of the phenotypic effect come to be amplified and to play a significant role in the direction that evolutionary change takes. This kind of directional-biasing effect (although it does not imply that evolution is “directed” in any vague intentional sense) cannot be addressed by natural selection theory itself.

Approaching evolutionary processes in a way that respects the complexity and self-organizing character of the substrates in which they occur can offer hints of a higher-order logic concerning nested levels of evolutionary processes within evolutionary processes. In the language evolution example, we can begin to see the essential interplay that must exist between self-organizing and selection processes that constitutes evolution. Not only must we recognize that evolutionary processes include self-organizing processes, but also that they may include nested levels of other evolutionary dynamics. To understand the details of language origins (including both the origins of biological language adaptations and the origins of linguistic structures) we need to understand how one evolutionary dynamic can emerge from another, i.e., how linguistic evolution can emerge from molecular evolution, and how this can feed back to alter the very nature of the antecedent process.

Language evolution was shown to be analogous to niche construction in some regards. Niche construction introduces a complicated compound interest effect into evolution in which products of behavioral intervention in the environment can feed forward to become selection pressures in future generations. What we might call the "linguistic environment" is a "niche" that is both a complex behavioral habit shared by a population of humans and yet also a set of highly specific cognitive constraints that places special demands on neural functioning. Though created by human activity the constraints that define it are not derived from human brains, but have emergent features derived from the demands of inter-individual communication and the demands of symbolic reference itself. These niche-like effects have likely shaped human brain evolution as much or more than any ecological demands in the past 2 million years.

In many ways, then, the niche construction model is far too narrowly conceived to capture the sort of phenotypic feed-forward effects that language entails. This analogy needs to be extended to include the special dynamism of linguistic evolution if it is to model brain-language coevolution accurately. Language and symbolic culture are not merely extrinsic inanimate products of human evolution. They are emergent, parasitic (or rather symbiotic), evolutionary processes whose independent dynamics (driven in part by semiotic factors never before relevant to biological evolution) have essentially become the tail that wagged the dog. These semiotic processes are active evolving systems that have their own self-organizing principles and evolutionary dynamics. These have been imposed on human evolutionary biology from the top down producing a totally unprecedented evolutionary dynamic. This evolutionary dynamic linking genes, brains, and symbolic culture is, in effect, a third-order evolutionary process. It is a level above Baldwinian or niche construction processes, both of which may be recruited in its service. In it, complex feed-forward effects predominate, amplifying their influence through an independent selection loop, making human bio-cultural evolution far more convoluted and unpredictable than anything biology has previously produced. There is, literally, no telling where it will lead! [For more detail, see refs. 5 and 8]

2. The explosion of interest in emergentism ("something more from nothing but") among both natural scientists and philosophers has not surprisingly generated considerable ambiguity in what is meant by the term emergence. I offer an inventory of emergent phenomena in the natural world, proposing that emergence takes three forms.

First-Order Emergence: Properties emerge as a consequence of shape interactions. Example: The interaction of water molecules (nothing but) generates a new property, surface tension (something more).

Second-Order Emergence: Properties emerge as a consequence of shape interactions played out over time, where what happens next is highly influenced by what has happened before. Example: The formation of a snowflake, where initial and boundary conditions become amplified in effect over time. In general, complex or "self-organizing" systems display second-order emergence.

Third-Order Emergence: Properties emerge as a consequence of shape, time, and "remembering how to do it." Example: Biology, where genetic and epigenetic instructions place constraints on second-order systems and thereby specify particular outcomes called biological traits. These traits then become substrates for natural selection by virtue of the fact that 1) their instructions are encoded and 2) they endow organisms with adaptive properties.

3. Though the special demands of communicating with symbols are in one sense abstract and formal, human neural biology has been made sensitive to them for the simple reason when these requirements are respected and embodied in culture they can produce incredible changes in the physical conditions of life and reproduction. This view of language evolution can be seen as a natural extension of developmental systems theory, but it is much more than this. It is no more than a glimpse into a much larger domain that has yet to be exposed: evolutionary semiotics. Until a more principled analysis of the logic of these sorts of processes is understood, we will remain in a prescientific phase of cognitive and social science, and lack a considerable part of the story of human brain evolution.

4. A vaguely similar reversal of Dawkins' view that memes are ideas and their external exemplars are something like the phenotype is presented by W. Benzon in his article "Culture as an evolutionary arena" (Journal of Social and Evolutionary Systems, Vol. 19, p. 321-362), though his justification of this is somewhat different.

5. Language, for example, didn't just evolve in the standard sense of that term. Language is an "emergent" biological phenomenon in ways unlike almost any other biological product [see ref. 7]. Many biologists have been forced to conclude, as I have, that language is as unprecedented in biology as was the very origins of the molecular genetic code. Imagining that explaining it would not require significant augmentation of existing evolutionary theory is, in my opinion, naive. Attempting to understand its emergent character by reducing it to a mere change in anatomy, an increase in intelligence, or a special case of computation inevitably obfuscates this central emergent nature. Because of this special status among evolved phenomena, the mystery of language forces us to explain it on different terms than say the evolution of upright posture or color vision. It is a problem analogous to explaining the evolution of an evolutionary process.

There are three additional considerations that I believe are indispensable to devising an adequate evolutionary approach to this mystery: (1) an appreciation of the role of self-organizing dynamics in the production of the complex systematicity recognized in language structures; (2) an evolutionary analysis that takes into account the multilevel interactions between biological evolutionary processes and non-biological linguistic evolutionary processes; and (3) an analysis of the semiotic infrastructure of language that is sufficient to articulate its ultimate functional constraints. Ultimately, the emergence of language offers a puzzle that requires us to rethink the concept of evolution itself in many significant ways, and demonstrates that it is intimately linked with processes of self-organization and semiosis [see 8]. One of the most important elements of this is the emergence of a new level of evolutionary dynamic out of biological evolution. With the emergence of language a second-order level of evolutionary process begins at the level of symbols themselves: linguistic evolution. This changes everything.

6. This is, of course, a bald statement with enormous implications that I cannot unfold here, but do deal with in depth in two upcoming books: Homunculus: How Brains make Experience and Golem: Making Things Think.

Graphics and Languages for the Flexible Modular Framework

John F. Sowa

Abstract. Every communication has syntax, semantics, and pragmatics. For computer communications, software designers have addressed syntax explicitly while leaving semantics and pragmatics implicit in their programs. But as software becomes more complex, the range of meanings (semantics) and purposes (pragmatics) grows without bounds. The failure to address purpose and meaning explicitly has led to different languages and GUIs for every conceivable purpose. By making meaning and purpose explicit, the designers can relate the bewildering variety of notations to a single semantic form: logic. Internal representations for logic can be any notation that is convenient for computer processing. External representations can be any form that people find convenient: graphics tailored for the applications, controlled versions of whatever natural languages the users speak, or any programming notations the developers happen to prefer. The unifying principle is the common logical form for both internal and external communications. To show how this principle can be implemented, this paper addresses the graphic and language interfaces for the Flexible Modular Framework (FMF) and their use in a semantically integrated development environment.

1 A Framework for Communication

The Flexible Modular Framework, as described by Sowa (2002), is an architecture for communication among interacting agents, which may be physical devices, computer programs, or human beings using any auditory, visual, or tactile interfaces. Computationally, the modules of an FMF may be specialized for any purpose whatever. Physically, they may be collected in a single box or be scattered anywhere across the Internet. Any module might have its own FMF nested within itself, and the nested modules could also be localized or distributed. The number of modules in an FMF may be growing and changing dynamically, and they may be accessed directly by a unique identifier or associatively according to the functions they perform. The communication protocol that relates the modules of an FMF emphasizes semantics and pragmatics, while allowing individual modules to use any syntax that may be appropriate for an application. This paper addresses the question of how a common logic-based protocol can support an open-ended range of human interfaces externally and computational notations internally.

The semantics of any computer program, including any module of the FMF, can be expressed in logic. Internally, the logic may be represented in any suitable notation, of which the most general and flexible is conceptual graphs (CGs). The human interface relates the logic to one of three forms:

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 31–51, 2004.

- **Controlled natural languages**, which are formally defined subsets of whatever natural language the user prefers. Examples in this paper are stated in Common Logic Controlled English (CLCE), but any other controlled NL that can be translated to and from logic may be used.
- **Application-oriented diagrams**, whose semantics is defined by the logic. If the users are software developers, the diagrams may be expressed in the Unified Modeling Language (UML), but there is no limit to the number or kinds of diagrams that may be used.
- **Multimedia resources**, which may be partly described in logic, but which cannot be fully represented in any symbolic notation. Examples include photographs, sounds, movies, and tactile simulations of three-dimensional structures.

Programmers who are familiar with computer-oriented languages and notations can use them when appropriate, but it is difficult for programmers to be fluent in multiple programming languages. For communications outside their core competency, even software professionals can use graphics and controlled NLs. By itself, pure first-order logic (FOL) can be used for only one purpose: to assert propositions. But Peirce (1904) observed that propositions can be used for many purposes other than making assertions:

> A proposition, as I have just intimated, is not to be understood as the lingual expression of a judgment. It is, on the contrary, that sign of which the judgment is one replica and the lingual expression another. But a judgment is distinctly more than the mere mental replica of a proposition. It not merely expresses the proposition, but it goes further and accepts it.... One and the same proposition may be affirmed, denied, judged, doubted, inwardly inquired into, put as a question, wished, asked for, effectively commanded, taught, or merely expressed, and does not thereby become a different proposition. (EP 2.311-312)

In natural languages, the purpose of a proposition can be expressed in several ways: sometimes by syntax, as in questions and commands; sometimes by the context of a message, a conversation, or an extended discourse; and sometimes by a complex statement with multiple nested statements. As an example, the sentence Tom believes that Mary wants to marry a sailor, contains three clauses, whose nesting may be marked by brackets:

```
Tom believes that [Mary wants [to marry a sailor]].
```

The outer clause asserts that Tom has a belief, which is expressed by the object of the verb believe. Tom's belief is that Mary wants a situation described by the nested infinitive, whose subject is the same person who wants the situation. Each clause states the purpose of the clause or clauses nested in it.

For logic to express the semantics and pragmatics of an English sentence, it must have an equivalent structure. The nested structure of the logic is shown explicitly in Figure 1, which is a conceptual graph for the sentence about Tom's belief. The large boxes, which contain nested CGs, are called contexts. The labels on those boxes indicate how the contexts are interpreted: what Tom believes is a proposition stated by the CG nested in the context of type Proposition; what Mary wants is a situation described by the proposition stated by the CG nested in the context of type Situation. The relations of type (Expr) show that Tom and Mary are the experiencers of states of believing or wanting, and the relations of type (Thme) link those states to the contexts that express their themes. The two relations attached to [Person: Mary] indicate that Mary is the experiencer of [Want] in one context and the agent (Agnt) of [Marry] in another context.

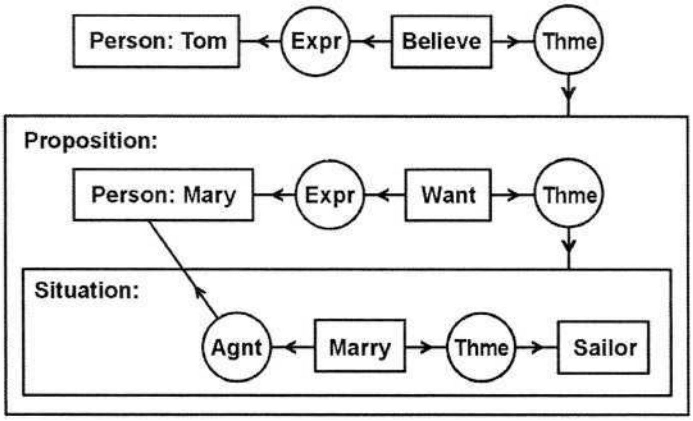


Fig. 1. A conceptual graph with nested contexts

When a CG is in the outermost context or when it is nested in a context of type Proposition, it states a proposition. When a CG is nested in a context of type Situation, the stated proposition describes the situation. When a context is translated to predicate calculus, the result depends on the type of context. In the following translation, the first line represents the subgraph outside the nested contexts, the second line represents the subgraph for Tom's belief, and the third line represents the subgraph for Mary's desire:

```
(∃a:Person)(∃b:Believe)(name(a,'Tom') ∧ expr(a,b) ∧ thme(b,
   (∃c:Want)(∃d:Situation)(person(Mary) ∧ expr(c,Mary)
     ∧ thme(c,d) ∧ dscr(d,
        (∃e:Marry)( ∃f:Sailor)(agnt(e,Mary) ?∧ thme(e,f)))))))
```

For the subgraph nested inside the context of type Situation, the description predicate dscr relates the situation d to the proposition expressed by the subgraph.

Each context of a CG or its translation to predicate calculus is limited in expressive power to pure first-order logic, but a proposition expressed in any context can make a metalevel assertion about the purpose or use of any nested proposition. To represent the model-theoretic semantics for a hierarchy of nested metalevels, Sowa

(2003) developed nested graph models (NGMs), which support a first-order style of model theory for each level and a first-order style of interconnections between levels. A hierarchy of metalevels with the NGM semantics can express the equivalent of a wide range of modal, temporal, and intentional logics. Equivalent hierarchies can also be expressed in controlled NLs by translating the syntactic structure of a complex sentence or discourse to the contexts of CGs or their equivalents in predicate calculus.

During the twentieth century, Peirce's observations about the various uses and purposes of propositions were independently rediscovered and elaborated by various philosophers and linguists. Each version focused on a different aspect of language use:

- **Language games.** Wittgenstein (1953) applied the term language game to a systematic way of using the vocabulary and syntax of a natural language to enable two or more people to collaborate or compete in a particular activity. Using the analogy of a chess game, he compared words to chess pieces, utterances to moves in the game, and purpose or pragmatics to the ultimate goal of winning the game. As examples, Wittgenstein gave more detail than Peirce:

  > Giving orders, and obeying them; describing the appearance of an object, or giving its measurements; constructing an object from a description (a drawing); reporting an event; speculating about an event; forming and testing a hypothesis; presenting the results of an experiment in tables and diagrams; making up a story, and reading it; play acting; singing catches; guessing riddles; making a joke, telling it; solving a problem in practical arithmetic; translating from one language into another; asking, thanking, cursing, greeting, praying.

- **Speech acts.** Austin (1962) developed a classification of speech acts as ways of using sentences in natural languages. The traditional three uses - declarative, interrogative, and imperative - are typically marked by syntax, but they can also be indicated by the verbs tell, ask, and command followed by a nested clause that states some proposition. Austin analyzed dozens of verbs, which he classified in five major categories: verdictives for passing judgment; exercitives for exercising authority; commissives for making commitments; behabitives for social behavior; and expositives for expository structure.
- **Discourse analysis.** Linguists have analyzed the structure of discourse in terms of anaphoric references, temporal sequence, argument structure, context, focus, and overall cohesion (Halliday and Hasan 1976). Of the various approaches, Rhetorical Structure Theory (Mann and Thompson 1987) has a high overlap with Austin's categories of speech acts, and it emphasizes the relationships among speech acts in the structure of the entire discourse.

These developments were not completely independent, since Wittgenstein had some acquaintance with Peirce's work (Nubiola 1996), Austin was familiar with Wittgenstein's work, and most linguists doing research on discourse analysis were familiar with Austin's work; but the connections among them have seldom been emphasized. When each approach is viewed as an aspect of the social processes underlying communication of any kind, they clarify and reinforce one another. Language games provide the structure that determines the purpose, the relevant topics, the appropriate speech acts, and the expected sequence for expressing them within the discourse.

In the FMF, any community of interacting agents can be viewed as the participants in a language game that motivates the interactions, both logical and physical, and the associated speech acts. For communications among the agents, McCarthy (1989) proposed a language called Elephant, which uses logic as the metalanguage for stating speech acts and as the object language for stating the propositional contents. The FMF incorporates a version of Elephant, in which the logic may be expressed in any suitable notation: controlled natural languages for communication with humans and any machine-oriented notation that may be appropriate for software agents. Graphics may also be used to supplement the logic for presenting relationships that are easier to show than to describe.

Section 2 of this paper surveys various notations for logic and their expression in controlled NLs. Section 3 presents a more detailed example of CLCE and ist use in describing a three-dimensional structure and its mapping to and from a relational database. Section 4 discusses tools and methods for integrating language and graphics. Section 5 discusses the use of logic and controlled NLs in a semantically integrated development environment. The concluding Section 6 shows how techniques for processing unrestricted natural languages can be used to detect and correct errors in controlled NLs and to help authors stay within the restrictions.

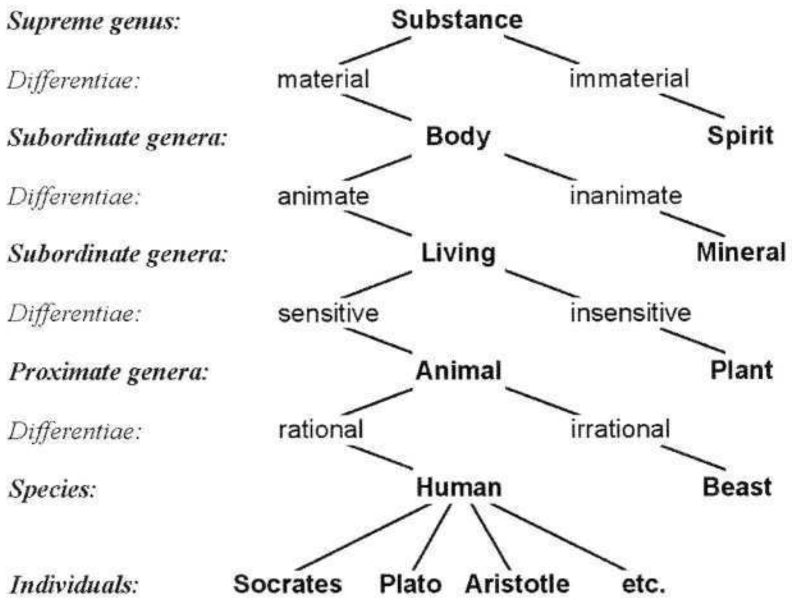
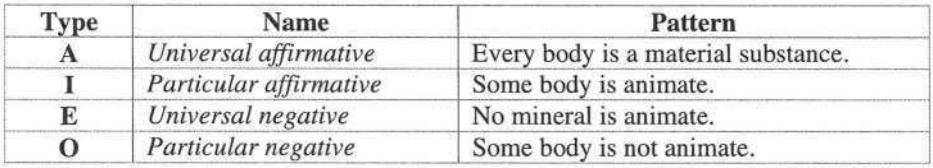


Fig. 2. Tree of Porphyry by Peter of Spain (1239)

2 Notations for Logic

For over two thousand years, controlled natural languages, supplemented with diagrams such as the Tree of Porphyry (Figure 2), were the only notations used for formal logic. In the late 19th century, the new mathematical notations for logic supported great technical advances. Unfortunately, they transformed logic from an introductory subject taught to every college freshman to an esoteric specialization known to an insignificant fraction of university graduates. With the development of computer science in the 20th century, syntax became a solved problem; the focus of research shifted to ontology and related semantic issues. Formal notations are still valuable for research, but for the practical problems of reading and writing knowledge representations, there is no reason why people should be forced to use notations designed for computer processing.

For his syllogisms, the first version of formal logic, Aristotle defined a highly stylized form of Greek, which became the world's first controlled natural language. In describing the relationships between Aristotle's logic and ontology, the philosopher Porphyry drew the first known hierarchy of categories. The version shown in Figure 2 was used in the middle ages to illustrate categories and their relationships to the syllogistic patterns, which were expressed in controlled Latin. Figure 3 shows the four types of propositions used in syllogisms and the sentence patterns that express them.



Type	Name	Pattern
A	*Universal affirmative*	Every body is a material substance.
I	*Particular affirmative*	Some body is animate.
E	*Universal negative*	No mineral is animate.
O	*Particular negative*	Some body is not animate.

Fig. 3. The four sentence patterns used in syllogisms

The sentence of type A indicates that a category such as Body is distinguished by the differentia material from its genus Substance. Sentences of types I, E, or O state implications or constraints on the hierarchy. The differentiae are the features or properties that distinguish a category from its supertypes, subtypes, or siblings. The hierarchy of types and subtypes of categories can be defined with the verb is. Differentiae and other properties can be expressed with any verb phrase, as in the following sentences:

```
Every human has two parents.
Every animal X can move some part of the body of X.
Every cat eats some food that is derived from an animal.
```

Although these sentence patterns may look like English, they are limited to a highly constrained syntax and semantics: each sentence has a fixed quantifier pattern, at most one negation, and a predicate that is true or false of the individuals indicated by the subject. This subset of English is sufficient to support the description logics, such as the Web Ontology Language (OWL), whose logical foundation is based on Aristotle's syllogisms. Every category or class of entities can be defined by one or more sentences of type A. Inheritance of properties from a type to an individual is determined by syllogisms of the following form:

```
Every human has two parents.
Socrates is a human.
Therefore, Socrates has two parents.
```

Other patterns determine inheritance from a supertype to a subtype, and syllogisms with negations (type E or O sentences) can check for inconsistencies. These sentence patterns can be mapped to specifications in many versions of logic. Following is the OWL representation for "`Every human has two parents`":

```
<owl:Class rdf:about="#Human">
  <rdfs:subClassOf><owl:Restriction>
  <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">2
     </owl:cardinality>
  <owl:onProperty rdf:resource="#Parent"/>
  </owl:Restriction></rdfs:subClassOf></owl:Class>
```

Another subset of logic is expressed by the Entity-Relationship diagrams, which are widely used in database design and software specifications. Figure 4 shows an E-R diagram that relates four entity types: `Student`, `Department`, `Course`, and `Section`. The entities are represented by boxes, and the relations by diamonds. Each path of box-diamond-box can be expressed by a pair of controlled English sentences, one for each direction of reading the path. The lower right corner of Figure 4, for example, may be expressed by the following two sentences:

```
Every section presents exactly one course.
Every course is taught in one or more sections.
```

As this example shows, the readings for the two directions may use different words, which need not occur inside the diamond node. However, the mapping from those words to the name of the relation must be specified by declaration statements for the chosen vocabulary. Those words are used in variants of type A sentences in which the second quantifier specifies a number or a range of numbers, such as `exactly one` or `at least 3 and no more than 7`.

The Unified Modeling Language (UML) includes type hierarchies similar to Figure 2, E-R diagrams similar to Figure 3, and several other kinds of diagrams used for software specifications. All UML diagrams can be expressed by controlled NL sentences that map to some subset of first-order logic (FOL). As an example, the UML activity diagrams, which are variants of Petri nets, can be expressed by the Horn-clause subset of FOL. Each activity or Petri-net transition can be specified by an `if-then` statement in which the `if`-part is a conjunction of preconditions, and the `then`-part specifies the activity and its postconditions:

```
If a copy of a book is checked out,
   the copy is returned,
   and a staff-member is available,
then the staff-member checks in the copy,
   the copy is available,
   and the staff-member is available.
```

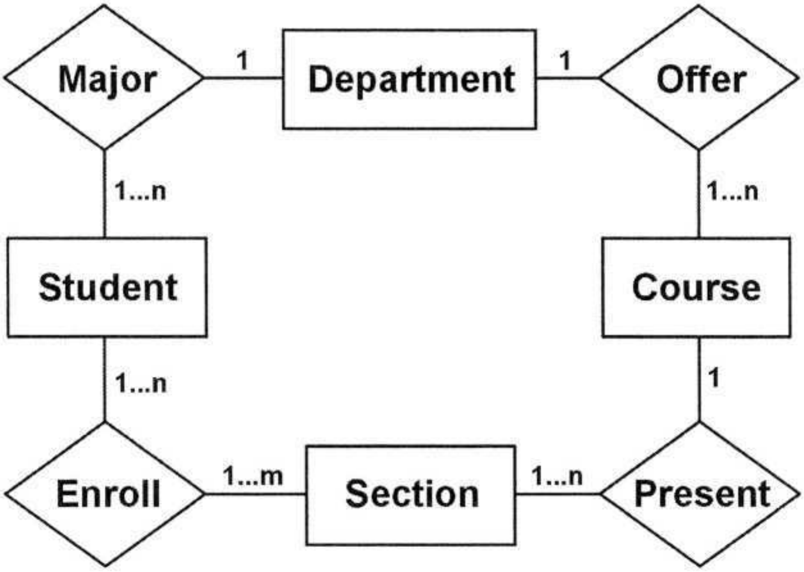


Fig. 4. E-R diagram for students, departments, courses, and sections

The three clauses of the `if`-part specify three preconditions. The clause immediately after the word `then` includes a verb that names the activity (check-in) and two noun phrases that identify its participants (the staff member and the copy). The final two clauses are the postconditions. Further details about the check-in activity could be specified by another activity diagram, Petri net, or paragraph of controlled English sentences. Sentences of this form are translated to executable form by the ACE system (Fuchs et al. 1999, Schwitter 1998). More recently, Fuchs, Schwitter, and others have extended the English syntax they support and the expressive power of the logic they generate.

All the controlled English examples in this paper can be expressed in Common Logic Controlled English (Sowa 2004). The CLCE translator maps CLCE to an internal representation in conceptual graphs, which it can then map to several notations for first-order logic: predicate calculus, Conceptual Graph Interchange Format (CGIF), and Knowledge Interchange Format (KIF). Following is the CGIF representation of the previous `if-then` sentence:

```
[If: (Copy [Copy: *x] [Book]) (CheckedOut ?x)
     (Returned ?x) (Available [StaffMember: *z])
   [Then: (CheckIn ?z ?x) (Available ?x) (Available ?z)] ]
```

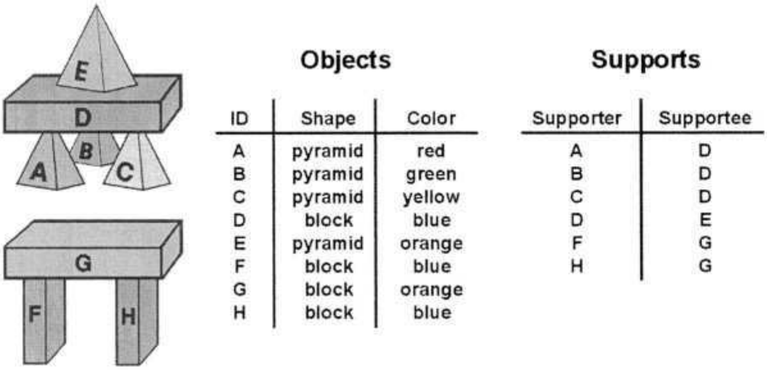
And following is the corresponding KIF statement:

```
(forall ((?x Copy)(?y Book)(?z StaffMember))
  (=> (Copy ?x ?y)(CheckedOut ?x)(Returned ?x)(Available ?y)
      (and (Checkin ?z ?x)(Available ?x)(Available ?y))))
```

CLCE, KIF, and CGIF can represent the full range of operators and quantifiers of first-order logic. By using FOL as the metalanguage, CLCE, KIF, and CGIF can be used to define and express extensions to FOL that can support versions of modal logic and higher-order logic. Gerbé and his colleagues (1998, 2000) used conceptual graphs as a metalanguage for software specifications, which are automatically translated to versions of controlled English and controlled French. Martin (2002, 2003) implemented a translator that maps controlled English to several notations, including UML, RDF, OWL, CGIF, and KIF.

3 Declaring the Syntax and Semantics of Words

Some examples in CLCE can be used to illustrate the mappings from controlled NLs to logic and related notations. Figure 5 shows two structures of blocks and pyramids and their representation in database tables named `Objects` and `Supports`. The first step is to declare the syntax of the words used in CLCE and relate them to the two database tables. Then CLCE descriptions of the structures can be automatically translated to logic, SQL, UML, or other notations.



Objects

ID	Shape	Color
A	pyramid	red
B	pyramid	green
C	pyramid	yellow
D	block	blue
E	pyramid	orange
F	block	blue
G	block	orange
H	block	blue

Supports

Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G

Fig. 5. Two structures represented in a relational database

For this example, CLCE nouns can be mapped to selections from the database tables, and CLCE names can be mapped to data in the tables themselves. In the following declaration, the syntax of each word is specified by its part of speech (e.g., noun or functional noun) and by a pattern for words that require more than one argument (e.g., x1 shape of x2). The semantics of each word is specified by the name of a predicate or relation that represents it in FOL and by an SQL query that maps the FOL relations to some selection from the database.

```
Declare object as noun
          from SQL("SELECT ID FROM OBJECTS"),
        shape as functional noun (x1 shape of x2)
          from SQL("SELECT SHAPE, ID FROM OBJECTS"),
        color as functional noun (x1 color of x2)
          from SQL("SELECT COLOR, ID FROM OBJECTS"),
        supporter as noun (x1 supporter of x2)
          with relation(support)
          from SQL("SELECT * FROM SUPPORTS"),
        supportee as noun (x2 supportee of x1)
          with relation(support);
```

The first line of the declaration states that the CLCE word object is a noun. Its default relation object is monadic when no variables are stated in the pattern, and the data that defines the relation is obtained from the SQL statement enclosed in parentheses. That statement extracts the ID column from the table named OBJECTS to define object(x). The next four lines define the words shape and color as functional nouns represented by dyadic relations, each of which corresponds to two columns of the OBJECTS table. The fifth line defines supporter as a relational noun, whose relation named support contains the complete data from the database table named SUPPORTS. The last line defines supportee as a relational noun with the same support relation, but with the arguments reversed.

The next declaration specifies all the entries in the database tables as CLCE names. In FOL, it is represented an existential quantifier for each name. Unlike variables, whose scope is limited to a single sentence, names have the entire text as scope.

```
Declare pyramid, block as name of shape,
        red, green, yellow, blue, orange as name of color,
        A, B, C, D, E, F, G, H as name of object.
```

After the nouns and names have been declared, the top structure shown in Figure 5 can be described by the following CLCE sentences. To illustrate the various stylistic options in CLCE, each object is described with a different stylistic convention.

```
The shape of A is pyramid; the color of A is red;
A is a supporter of D.

Pyramid is the shape of B; green is the color of B;
a supporter of D is B.

C has pyramid as shape; C has yellow as color;
D is a supportee of C.
```

```
D is an object that has block as shape;
the object D has blue as color;
a supporter of the supportee E is the supporter D.

The shape of E is pyramid, and the color of E is orange.
```

Different stylistic choices in CLCE may lead to different representations in FOL, but they are all logically equivalent. Much of the variation is reduced or eliminated in the translation from CLCE to FOL, and the FOL rules of inference can be used to determine the equivalence of any remaining variability. The CLCE translator would map the above sentences to ground-level assertions in any notation for FOL, and the tools that map to SQL would generate database updates.

The declaration statements above defined nouns and names that were sufficient to describe the two structures of blocks and pyramids; no verbs, adjectives, or prepositions are needed. However, a logically equivalent description could be stated by representing the database table `Supports` by the English verb `support`. The following declaration defines that verb with two patterns for expressing relations in either active voice or passive voice:

```
Declare support as verb (instrument supports theme)
       (theme is supported by instrument)
       from SQL("SELECT * FROM SUPPORTS");
```

In the active pattern, the role `instrument (Inst)`, which comes before the verb, occurs in subject position, and the role `theme (Thme)`, which comes after the verb, occurs in object position. In the passive pattern, those two positions are reversed: the theme is the subject, and the instrument is the object of the preposition by. Those two roles are used in the role representation for verbs; the presence of two roles in the declaration indicates that the relation is dyadic. The SQL statement indicates the source of the data; it may be omitted if the relation `support(x1,x2)` has already been specified by an earlier declaration.

With the verb `supports` many propositions, such as `A supports B`, can be stated more simply than with the nouns `supporter` and `supportee`. But for some propositions, the nouns can be used to simplify the statements:

```
For every object x that has orange as color,
every supporter of x has block as shape.

For every object x that has orange as color,
every object that supports x has block as shape.
```

Instead of using the word `block` as the name of a shape and `orange` as the name of a color, it may be more convenient to declare `block` as a special kind of object and to declare `orange` as an adjective. Those declarations would require more complex SQL statements to define the monadic predicates that correspond to the noun and the adjective:

```
Declare block as noun
       from SQL("SELECT ID FROM OBJECTS
          WHERE SHAPE='block'"),
     orange as adjective
       from SQL("SELECT ID FROM OBJECTS
          WHERE COLOR='orange'").
```

With these declarations, the previous statements become

```
For every orange object x,
every supporter of x is a block.

For every orange object x,
every object that supports x is a block.
```

The current version of CLCE prohibits universally quantified noun phrases from being used as the object of a preposition. Without that restriction, the previous statement could be simplified further:

```
Every supporter of every orange object is a block.
```

These examples show how CLCE can be related to a relational database by means of the keyword `SQL` in the declarations. The SQL syntax is not recognized by the CLCE translator, which merely passes the quoted SQL statement to an interface program that links CLCE to the database. Other keywords, such as `UML` or `URI`, could be used to link CLCE to data from UML definitions or from resources located anywhere on the Internet.

4 Relating Language and Graphics

A picture may be worth a thousand words, but a few words can often explain a thousand pictures. Silent movies used text to explain the action, and as soon as sound became available, the silent movies became obsolete. Language and graphics are both important, and good tools are needed to integrate them. The central requirement for integration is a common logical representation, but a wide range of tools are needed to relate language and graphics, translate them to and from logic automatically, and help analyze informal versions of natural language and graphics.

Translating informal graphics to logic is as difficult as translating unrestricted natural language to logic, but it is much easier to translate logic or a controlled NL to graphics. Cyre and his students (1994, 1997, 2001) have developed tools and techniques for analyzing both the language and the diagrams of patent applications and translating them (semiautomatically) to conceptual graphs. They also designed a scripting language for automatically translating CGs to circuit diagrams, block diagrams, and other graphic depictions. Their tools can also translate CGs to VHDL, a hardware design language used to specify very high-speed integrated circuits (VHSIC). To add comments to the VHDL specifications, they developed the Controlled English Commenter (CEC) for helping human designers write controlled

English and stay within its restrictions. After writing a few comments and being corrected and guided by the CEC, the users quickly learned to adapt to the CE restrictions and stay within its limited syntax and vocabulary.

For Cyre's constructions, one kind of graph is translated to another: some concepts of a CG map to icons; other concepts specify modifiers that may change the size, shape, color, or other properties of the icons; and relations specify various connections among the icons and their properties. Three-dimensional scenes allow a greater range of variability, which require a considerable amount of background knowledge for determining how objects are related. The WordsEye system (Coyne & Sprout 2001) analyzes descriptions stated in controlled English, translates them to a logical form, constructs a 3D scene containing the objects mentioned, and maps the scene to a 2D display. Following is the description from which WordsEye generated a scene and displayed it:

```
John uses the crossbow. He rides the horse by the
store. The store is under the large willow. The small
allosaurus is in front of the horse. The dinosaur
faces John. A gigantic teacup is in front of the
store. The gigantic mushroom is in the teacup. The
castle is to the right of the store.
```

For each object mentioned, WordsEye has a default representation. Adjectives such as small, large, and gigantic modify the representations, and verbs such as uses and rides relate the objects and adapt their representations appropriately. As the authors admit, WordsEye will not replace other methods for creating scenes, but even in its current form, it is a quick way of setting up a scene that can be refined by other graphics tools.

For software development, many visual tools allow programmers to generate applications by drawing lines between predefined modules. But a combination of graphics and controlled NL can be used to design, specify, and implement every aspect of a system at any level of detail. Petri nets are one of the most general graphics tools, and they have been used successfully to design and implement complex network applications with distributed interactions among multiple agents. Yet every link and node of a Petri net can be specified in the Horn-clause subset of FOL, and every Horn-clause can be written as an `if-then` sentence in controlled English, as illustrated in Section 2. A combination of Petri nets with controlled NLs can provide a complete programming language. Programmers should have the option of using conventional programming languages, but they could also create or modify modules by graphic manipulation or by statements in a controlled NL, and the system could respond by explaining any node or link by a comment in the same NL. Any declarations or details that cannot be shown graphically can be stated in a controlled NL.

Under the name of activity diagrams, Petri nets are one of the diagrams used in the Unified Modeling Language. The UML diagrams specify information at one of four metalevels: the metametalanguage defines the syntax and semantics of the UML diagrams; the metalanguage defines the general-purpose UML types; a systems analyst defines application types as instances of the UML types; finally, the working data of an application program consists of instances of the application types. To provide a unified view of all these levels, Gerbé and his colleagues (1998) implemented design tools that use conceptual graphs as the representation language at every level. For his PhD dissertation, Gerbé (2000) developed an ontology for using CGs as the metametalanguage for defining CGs themselves. He also applied it to other notations, including the UML diagrams and the Common KADS system for designing expert systems. Using that theory, Gerbé and his colleagues developed the Method Repository System as an authoring environment for editing, storing, and displaying descriptions of business rules and processes. Internally, the knowledge base is stored in conceptual graphs, but externally, the graphs can be translated to web pages in either English or French. About 200 business processes were modeled in a total of 80,000 CGs.

5 Semantically Integrated Development Environment

A smooth migration path is essential to ease the introduction of any new technology. Many excellent languages and tools have languished because they made an abrupt break with the past without supporting a period of peaceful coexistence of old and new. The Flexible Modular Framework is designed to support coexistence by providing "wrappers" that enable existing hardware and software modules to participate as first-class citizens in an FMF. The transition to a logic-based technology requires similar support for human users and developers who never studied logic. Fortunately, every human speaks a natural language, and the syntax and semantics of every NL contains a subset of first-order logic. Furthermore, many technologies that are familiar to programmers and systems analysts are also based on subsets of FOL: UML diagrams for software specifications, the SQL database language, the Express language for manufacturing specifications, the RDF and OWL languages for the semantic web, and many specialized notations for metadata and metalanguage specifications. All these languages and notations can be translated to and from logic for computer processing and controlled NLs for human consumption.

Widespread acceptance of any new language or interface occurs when the total effort for application design, development, and deployment is significantly reduced in comparison to more familiar methods. Over the past thirty years, many natural-language query systems (which could more accurately be called controlled NL systems) were developed, and they were generally much easier to use than SQL. The major stumbling block that has prevented them from becoming commercially successful is the amount of effort required to define the vocabulary terms and map them to the appropriate database fields. If that effort is added on top of the application design and development, acceptance will be slow. If it requires trained linguists or even people who remember their high-school grammar lessons, acceptance will be nearly impossible.

For controlled NLs to be successful, the tools for building applications based on them must be easy to use by current software developers, and they must accommodate legacy applications based on older technologies. One way to achieve that goal is to give the developers an interface that is as easy to use as the ones they design for their own users. Such an interface should be supported by the following kinds of tools and resources:

1. **Dictionaries.** The kinds of syntactic declarations illustrated in Sections 2 and 3 should be derived from predefined dictionaries and terminologies. For specialized terms, such as `staff-member`, the tools could suggest the syntactic features of `member`, and let the developer accept them or modify them as needed.

2. **Starting ontologies.** Despite the many ontology projects that have been active during the past decade, no standardized ontologies exist, and no universally accepted ones are likely to appear any time soon. Furthermore, totally new ontologies are guaranteed to be incompatible with legacy applications. For a long time to come, ontologies will have to evolve from the current mix of vocabularies, terminologies, and libraries of ad hoc piecemeal solutions. New tools must be able to use and build upon the older resources.

3. **Mappings to existing tools.** New tools must coexist with old tools. One approach is to design new ones as plug-ins that supplement the tools designed for older notations with a more readable alternative. One example would be to supplement the popular UML diagrams with tools that can automatically translate any link or subgraph to and from logic or controlled NLs. No developers would be forced to change overnight, but all developers could benefit from the use of controlled NLs for a gradually increasing range of applications.
4. **Legacy re-engineering.** The best time to convert to a new system is when managers have already decided that the old system is inadequate. At that point, a major conversion effort is underway, and tools that can analyze both the old implementation and old documentation are needed. If those tools generate controlled NLs as a supplement to the documentation, they can generate the dictionaries and ontologies needed for the new application as a byproduct.

5. **Integrated interface.** The ultimate goal is to make a controlled NL the primary language for all human-computer interactions and to use it in both written and spoken forms. Graphics will also be important, but controlled NLs are necessary for help, explanations, and complex information that cannot be expressed in menus and icons. Anybody who knows traditional computer languages can use them for niche applications or for working with internal computer processes. But computer experts are also human, and they can benefit from using their own native language when appropriate.

A very important side effect of using controlled NLs as the implementation language for computer applications is that the documentation and the implementation become identical. Every change to the documentation also updates the implementation, and no discrepancy between them is possible.

Completely new tools can be introduced for purposes that have not been handled adequately by older tools. An important example is the task of analyzing, editing, and translating documentation to a computable form. Skuce and his colleagues (2000) designed tools that help an editor select, analyze, and translate unrestricted natural language to a controlled NL called ClearTalk. The knowledge editing (KE) tools for writing controlled NLs have the following advantages over specialized notations for defining ontologies, rules, and other knowledge representations:

- **Reduced training.** Anyone who can read English can immediately read ClearTalk, and the people who write ClearTalk learn to write it while using it. The ClearTalk system itself does most of the training through use: the restrictions are shown by menus and templates and are enforced by immediate syntactic checks. By consistently using ClearTalk for its output, the system reinforces the acceptable syntactic forms.

- **Reduced complexity.** During the knowledge editing process, KE tools detect ambiguities in the source documents and help the human user select relevant passages and clarify or correct vague passages. After the knowledge has been translated to ClearTalk with human assistance, the restricted syntax of ClearTalk eliminates the syntactic ambiguities of ordinary language. The semantic ambiguities are eliminated by the system, which enforces a single definition for every term. As a result, the system can automatically translate ClearTalk to and from logic and other computer-oriented languages.
- **Self-documenting.** People can read ClearTalk without special training, and a computer system can translate it automatically to an executable form. As a result, the comments and the implementation become identical, and there is never a discrepancy between what the human reads and what the machine is processing.
- **Annotations.** The ClearTalk generated by the KE tools can also be written as an annotation to the original source documents. Those annotations can serve as humanly readable comments or as input to other tools that process ClearTalk.

As an example, the students in Skuce's operating systems course used the KE tools to map information from on-line Linux manuals to a knowledge base for a Linux help facility. The people who wrote the manuals were experts, but the students who edited the knowledge base were novice users of both Linux and the KE tools. Skuce and his colleagues developed a system called FactGuru, which organizes the knowledge base, presents it in a convenient form, and helps users find and use the knowledge.

Even IT professionals need help in dealing with the proliferation of new languages and notations for every aspect of their jobs. The tools that are supposed to help them often add to their burden, as one systems programmer expressed in a poignant cry:

> Any one of those development tools, by itself, can be a tremendous aid
> to productivity, but any two of them together can kill you.

No one should be required to learn different languages for the database, the ontologies, the web pages, the programming, the servers, the clients, the network, and the open-ended variety of "helpful" development tools. Everything can be done with the syntax and vocabulary of the user's native language, supplemented with appropriate graphics for each aspect of the process. Any expert who prefers to use a more specialized language for some aspect of software development is welcome to use it, but nobody can be an expert in every aspect simultaneously.

By itself, a controlled NL solves only the syntactic part of the problem, which is not the most difficult aspect of learning any programming system. Even harder is learning the names, icons, or menu locations that somebody chose for every feature. What one person calls a directory, another will call a folder. One person says "import", and another says "include"; "bookmarks" or "favorites"; "branch" or "jump"; "network" or "graph"; "call", "perform", "execute", or "do"; "subroutine", "procedure", "function", or "method". Sometimes these terms are synonyms, sometimes they mark important distinctions, and sometimes different people distinguish them differently. Standardized terminologies are useful, but new terms are constantly being invented, old terms become obsolete, and the total number of terms grows beyond anyone's capacity to learn, remember. or use. The semantic structures and behaviors associated with those terms are still harder to learn. Finally, the hardest to learn and most important of all is knowing how to use these things and why. The fundamental issues are semantics and pragmatics.

6 Using Structure to Interpret Language

Natural languages are often called unstructured, but it is more accurate to say that NLs can express such a wide range of structures that it is difficult for a computer program to detect which structure is being expressed by any particular phrase. In effect, An unstructured NL is the totality of all language games that can be played with a given syntax and semantics. A controlled NL is easier to process because it is constrained to one or at most a small number of predefined language games. Each language game has an associated set of semantic structures, which determine what can be said and how each statement relates to any other. Being able to retrieve the right structures at the right time is essential for both computers and humans to understand the language and play the game.

The techniques for finding and using the correct semantic structures, which are essential for understanding unrestricted NLs, can also be used to assist users who stray outside the boundaries of a controlled NL. As an example, Sowa and Majumdar (2003) showed how the Intellitex parser and the VivoMind Analogy Engine (VAE) were able to retrieve semantic structures and use them during the process of language interpretation. In one major application, LeClerc and Majumdar (2002) used Intellitex and VAE to analyze both the programs and the documentation of a large corporation, which had systems in daily use that were up to forty years old. Although the documentation specified how the programs were supposed to work, nobody knew what errors, discrepancies, and obsolete business procedures might be buried in the code.

The task, called legacy re-engineering, required an analysis of 100 megabytes of English, 1.5 million lines of COBOL programs, and several hundred control-language scripts, which called the programs and specified the data files and formats. Over time, the English terminology, computer formats, and file names had changed. Some of the changes were caused by new computer systems and business practices, and others were required by different versions of federal regulations. The requirements were to analyze any English text or programming code that referred to files, data, or processes in any of the three languages (English, COBOL, and JCL), to generate an English glossary of all process and data terminology, to define the specifications for a data dictionary, to create UML diagrams of all processes and data structures, and to detect inconsistencies between the documentation and the implementation.

To understand the power and limitations of the Intellitex parser and semantic interpreter, it is important to realize that Intellitex cannot, by itself, translate informal English to executable programs. That possibility was dismissed by the pioneer in computer science Alan Perlis, who observed “It is not possible to translate informal specifications to formal specifications by any formal algorithm.” English syntax is not what makes the translation difficult. The difficulty arises from the enormous amount of background knowledge that lies behind every word in English or any other natural language.

But Intellitex was not used to translate informal English to formal conceptual graphs. Instead, Majumdar first used it to analyze the formal specifications encoded in the programs and the database. Those unambiguous specifications were translated to conceptual graphs and became the semantic structures for interpreting the English documentation. When Intellitex processed English sentences, it used the previously generated CGs to resolve ambiguities and to provide the necessary background knowledge. As an example, the following paragraph is taken from the documentation:

> The input file that is used to create this piece of the Billing Interface for the General Ledger is an extract from the 61 byte file that is created by the COBOL program BILLCRUA in the Billing History production run. This file is used instead of the history file for time efficiency. This file contains the billing transaction codes (types of records) that are to be interfaced to General Ledger for the given month. For this process the following transaction codes are used: 32 - loss on unbilled, 72 - gain on uncollected, and 85 - loss on uncollected. Any of these records that are actually taxes are bypassed. Only client types 01 - Mar, 05 - Internal Non/Billable, 06 - Internal Billable, and 08 - BAS are selected. This is determined by a GETBDATA call to the client file. The unit that the gain or loss is assigned to is supplied at the time of its creation in EBT.

Most of the words in this paragraph are found in the VivoMind dictionary, which is based on WordNet with numerous additions and extensions. Many other words, however, are not found, such as BILLCRUA, GETBDATA, and EBT. In isolation, this paragraph would be difficult for a human to understand. However, this paragraph did not appear in isolation. The background knowledge necessary to interpret most of the unknown words was found by first processing the COBOL and JCL programs. The names, types, and interrelationships of all the files, programs, data structures, and variables were found in those programs. As Intellitex processed the COBOL and JCL programs, it added those names to its dictionary along with their types and the CGs that represented their relationships to other data. When it processed English, Intellitex used that information to resolve ambiguities and to relate information from different sources. This task is sometimes called knowledge fusion.

This example also illustrates how Intellitex can process a wide range of syntactic constructions with a rather simple grammar. A phrase such as “32 - loss on unbilled” is not covered by any published grammar of English. When Intellitex found that pattern, it did not reject it; instead, it translated it to a rudimentary conceptual graph that linked the concept `[Number: 32]` by an unknown relation to a CG of the following form:

```
[Loss]?(On)?[Unbilled]
```

The result was stored as a tentative interpretation with a low weight of evidence. But Intellitex soon found two more phrases with the same syntactic pattern: "72 - gain on uncollected" and "85 - loss on uncollected". Therefore, Intellitex assumed a new grammar rule for this pattern, gave a name to the unknown relation, and associated it with the new grammar. By using VAE to find analogies to the CGs found in the COBOL programs, Intellitex discovered the particular COBOL program that defined the unknown relation, and it verified that 32, 72, and 85 were transaction codes assigned to subprocedures in that program. Although that syntactic pattern is not common in the full English language, it is important for the analysis of at least this one document. Such patterns, which may be called nonce grammar, often occur in specialized sublanguages of technical English, as used in business, law, medicine, science, engineering, and the military.

In three weeks of computation on a 750 MHz Pentium III, VAE combined with the Intellitex parser was able to analyze all the documentation and programs and generate one CD-ROM containing the required results. Several factors enabled Intellitex to process unrestricted English:

1. The unambiguous COBOL and JCL programs were analyzed first in order to provide Intellitex with the semantic structures necessary to interpret relevant information in the English documentation.

2. Any English sentences that did not match anything found in the COBOL or JCL were discarded as irrelevant.

3. The high-speed analogy engine enabled the relevant semantic structures to be accessed and used during the process of parsing and interpreting the English sentences. Any ambiguities were resolved immediately by comparing tentative parses to the expected semantic connections.

4. Unusual syntactic patterns that were not covered by the grammar were accepted if their constituents matched semantic patterns found in the COBOL and JCL. Patterns that occurred repeatedly acquired the status of new grammar rules.

In effect, the first stage of processing the COBOL and JCL defined the semantics of a new language game, which enabled the documentation to be processed as if it were a special kind of controlled English.

The method of using VAE to find the semantic structures needed for interpreting natural language can also support a powerful help facility for correcting sentences that fall outside the boundaries of a controlled NL. By supplementing a controlled NL parser with Intellitex and VAE, the KE tools could support a two-level system:

1. **Syntactic correction.** The first level would be an ordinary parser for the controlled NL. The usual error-processing methods could detect and correct

many simple errors in spelling and syntax. This level of processing would be sufficient for most routine errors.

2. **Semantic correction.** If the first-level parser was unable to correct the error, it could invoke Intellitex and VAE to find an appropriate semantic structure that might be used to interpret the input and suggest a possible semantic correction. For example, a programmer who was switching from Windows to Linux might specify the wrong options for some command. By using analogies, VAE might be able to find a recommended alternative from the Linux manuals or present a menu of the correct options.

A two-level system of this sort would blur the boundary between controlled and unrestricted NLs. The first-level parser would handle the inputs that are syntactically correct or can be corrected by minor adjustments. The second-level parser would use the techniques developed for unrestricted NLs to find background knowledge that enable semantic corrections, suggest alternatives, or lead to more extensive help and tutoring facilities. The user would experience a more forgiving system that would make context-sensitive suggestions instead of rejecting an incorrect input.

References

1. Austin, John L. (1962), How to do Things with Words, second edition edited by J. O. Urmson & Marina Sbisá, Harvard University Press, Cambrige, MA, 1975.
2. Coyne, Bob, & Richard Sproat (2001) “WordsEye: An automatic text-to-scene conversion system,” Proc. SIGGRAPH 2001, Los Angeles.
3. Cyre, W. R., S. Balachandar, & A. Thakar (1994) “Knowledge visualization from conceptual structures,” in Tepfenhart et al. (1994) Conceptual Structures: Current Practice, Lecture Notes in AI 835, Springer-Verlag, Berlin, pp. 275-292.
4. Cyre, W. R. (1997) “Capture, integration, and analysis of digital system requirements with conceptual graphs,” IEEE Trans. Knowledge and Data Engineering, 9:1, 8-23.
5. Cyre, W. R., & P. Victor (2001) “An intelligent interface for model documentation with controlled english,” Proc. IASTED 2001, Pittsburgh.
6. Fuchs, Norbert E., Uta Schwertel, Rolf Schwitter (1999) Attempto Controlled English (ACE), Language Manual, Technical Report ifi-99.03, University of Zurich. For further detail, see http://www.ifi.unizh.ch/attempto/
7. Gerbé, Olivier, R. Keller, & G. Mineau (1998) “Conceptual graphs for representing business processes in corporate memories,” in M-L Mugnier & Michel Chein, eds., Conceptual Structures: Theory, Tools, and Applications, LNAI 1453, Springer-Verlag, Berlin, pp. 401-415.
8. Gerbé, Olivier (2000) Un Modèle uniforme pour la Modélisation et la Métamodélisation d’une Mémoire d’Entreprise, PhD Dissertation, Département d’informatique et de recherche opérationelle, Université de Montréal.
9. Halliday, M.A.K. and R. Hasan (1976) Cohesion in English, Longman, London.
10. Harris, Roy (1988) Language, Saussure, and Wittgenstein: How to Play Games with Words, Routledge, London.
11. Mann, William C., & Sandra A. Thompson (1987) “Relational Propositions in Discourse,” Discourse Processes 9:1, pp. 57-90.
12. Martin, Philippe (2002) “Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English,”

13. Martin, Philippe (2003) "Translations between UML, OWL, KIF, and the WebKB-2 languages (For-Taxonomy, Frame-CG, Formalized English)," http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/mode/comparisons.html
14. McCarthy, John (1989) "Elephant 2000: A programming language based on speech acts," http://www-formal.stanford.edu/jmc/elephant.html
15. Nubiola, Jaime (1996) "Scholarship on the relations between Ludwig Wittgenstein and Charles S. Peirce," in I. Angelelli & M. Cerezo, eds., Proceedings of the III Symposium on the History of Logic, Gruyter, Berlin.
16. Peter of Spain or Petrus Hispanus (circa 1239) Summulae Logicales, edited by I. M. Bochenski, Marietti, Turin, 1947.
17. Schwitter, Rolf (1998) Kontrolliertes Englisch fúr Anforderungsspezifikationen, Studentdruckerei, Zurich. For further references, see http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages/
18. Skuce, Doug (2000) "Integrating web-based documents, shared knowledge bases, and information retrieval for user help," Computational Intelligence 16:1.
19. Sowa, John F. (2000) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole, Pacific Grove, CA.
20. Sowa, John F. (2002) "Architectures for Intelligent Systems," IBM Systems Journal 41:3, pp. 331-349.
21. Sowa, John F. (2003) "Laws, facts, and contexts: Foundations for multimodal reasoning," in V. F. Hendricks et al., eds., Knowledge Contributors, Kluwer, Dordrecht, pp. 145-184.
22. Sowa, John F., & Arun K. Majumdar (2003) "Analogical reasoning," in A. de Moor, W. Lex, & B. Ganter, eds., Conceptual Structures for Knowledge Creation and Communication, LNAI 2746, Springer-Verlag, Berlin, pp. 16-36.
23. Sowa, John F. (2004) "Common Logic Controlled English Specifications," http://www.jfsowa.com/clce/specs.htm
24. Wittgenstein, Ludwig (1953) Philosophical Investigations, Basil Blackwell, Oxford.

Implicational Concept Graphs

Rudolf Wille

Technische Universität Darmstadt, Fachbereich Mathematik,
Schloßgartenstr. 7, D–64289 Darmstadt
wille@mathematik.tu-darmstadt.de

Abstract. This paper introduces *implicational concept graphs* as special existential concept graphs of power context families and shows how such implicational concept graphs give rise to a *mathematical semantics of implications.* The advantage of the offered mathematical semantics is that it opens the door to mathematical structure theory with its rich structural insights and patterns of argumentations. As a consequence, it could be proved that the *implicational theory* of implicational concept graphs is equivalent (in the main) to the theory of attribute implications of formal contexts. This result could even be generalized to an analogue result for *clausal concept graphs.*

1 A Mathematical Semantics of Implications

Implications are in the heart of logics. Therefore implications should have a central status in Contextual Logic too. Since *Contextual Logic* is understood as a mathematization of traditional philosophical logic with its doctrines of concepts, judgments, and conclusions [Wi00], in Contextual Logic, implications as special judgments have to be semantically determined as mathematized judgments, i.e., as specific concept graphs of power context families. But how to obtain such semantical representation of implications? Our solution is guided by the observation that there are always some variables implicitly or even explicitly involved in linking the premise of an implication to its conclusion. Therefore implications are mathematized in this paper as special existential concept graphs [Wi02] which are the union of two designated subgraphs playing the role of premise and conclusion; such existential concept graphs are called "*implicational concept graphs*".

The advantage of such a mathematical semantics of implications is that it opens the door to mathematical structure theory with its rich structural insights and patterns of argumentations. This is demonstrated in Section 4 by proving two propositions which have as consequence that the *implicational theory of implicational concept graphs* of power context families is equivalent (in the main) to the theory of attribute implications of formal contexts (cf. [GW99b]). In Section 5, the implicational concept graphs are generalized to *clausal concept graphs* whose theory can be linked to the theory of attribute clauses of formal contexts. This again opens connections to known mathematical structure theory which substantially enriches Contextual Judgment Logic. An example of applying the Contextual Logic semantics of implications will be presented in [Wi04].

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 52–61, 2004.

2 Concept Graphs of Power Context Families

The proposed Contextual Logic semantics for implications is based on notions and results of *Contextual Judgment Logic* which shall be recalled from [Wi02] to make this paper more easy to read. For basic notions and results from *Formal Concept Analysis,* the reader is referred to [GW99a].

A *power context family* is a sequence $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ of formal contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ with $G_k \subseteq (G_0)^k$ for $k = 1, 2, \ldots$. The formal concepts of $\mathbb{K}_k$ with $k = 1, 2, \ldots$ are called *relation concepts,* because they represent k-ary relations on the object set G_0 by their extents.

A *relational graph* is a structure (V, E, ν) consisting of two disjoint sets V and E together with a map $\nu : E \to \bigcup_{k=1,2,\ldots} V^k$; the elements of V and E are called *vertices* and *edges,* respectively, and $\nu(e) = (v_1, \ldots, v_k)$ is read: $v_1, \ldots, v_k$ are the *adjacent vertices* of the *k-ary edge* e ($|e| := k$ is the *arity* of e; the arity of a vertex is defined to be 0). Let $E^{(k)}$ be the set of all elements of $V \cup E$ of arity k ($k = 0, 1, 2, \ldots$).

A *concept graph* of a power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ with $\mathbb{K}_k := (G_k, M_k, I_k)$ for $k = 0, 1, 2, \ldots$ is a structure $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ for which

- (V, E, ν) is a relational graph,
- $\kappa: V \cup E \to \bigcup_{k=0,1,2,\ldots} \underline{\mathfrak{B}}(\mathbb{K}_k)$ is a mapping such that $\kappa(u) \in \underline{\mathfrak{B}}(\mathbb{K}_k)$ for all $u \in E^{(k)}$,
- $\rho: V \to \mathfrak{P}(G_0)\backslash\{\emptyset\}$ is a mapping such that $\rho(v) \subseteq Ext(\kappa(v))$ for all $v \in V$ and, furthermore, $\rho(v_1) \times \cdots \times \rho(v_k) \subseteq Ext(\kappa(e))$ for all $e \in E$ with $\nu(e) = (v_1, \ldots, v_k)$; in general, $Ext(\mathfrak{c})$ denotes the extent of the formal concept $\mathfrak{c}$.

It is convenient to consider the mapping ρ not only on vertices but also on edges: for all $e \in E$ with $\nu(e) = (v_1, \ldots, v_k)$, let $\rho(e) := \rho(v_1) \times \cdots \times \rho(v_k)$.

For illustrating notions and results presented in this paper, a power context family $\vec{\mathbb{K}}(\mathbb{N})$ shall be considered which have

- $\mathbb{N} := \{1, 2, 3, 4, 5, \ldots\}$ as its object set G_0,
- even, odd, prim, square, etc. in its attribute set M_0,
- less than 10, greater than 100, etc. in its attribute set M_1,
- successor of, less than, equal to, divides, etc. in its attribute set M_2,
- between, sum up to, greatest common divisor of, etc. in its attribute set M_3.

Fig.1 shows some examples of concept graphs of the power context family $\vec{\mathbb{K}}(\mathbb{N})$ which are drawn according to Sowa's convention of representing conceptual graphs (cf. [So84]).

A *subgraph* of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ is a concept graph $\mathfrak{G}_s := (V_s, E_s, \nu_s, \kappa_s, \rho_s)$ for which $V_s \subseteq V$, $E_s \subseteq E$, $\nu_s = \nu|_{E_s}$, $\kappa_s = \kappa|_{V_s \cup E_s}$, and $\rho_s = \rho|_{V_s}$. In Fig.1, the concept graph (1) is obviously a subgraph of the concept graph (3).

The *union* and *intersection* of subgraphs $\mathfrak{G}_t := (V_t, E_t, \nu_t, \kappa_t, \rho_t)$ $(t \in T)$ of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ is defined by

$$\bigcup_{t \in T} \mathfrak{G}_t := (\bigcup_{t \in T} V_t, \bigcup_{t \in T} E_t, \bigcup_{t \in T} \nu_t, \bigcup_{t \in T} \kappa_t, \bigcup_{t \in T} \rho_t),$$

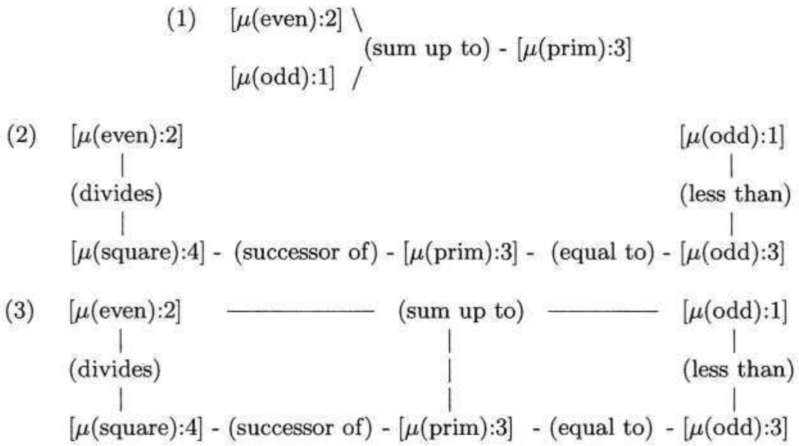


Fig. 1. Concept graphs of the power context family $\vec{\mathbb{K}}(\mathbb{N})$

$$\bigcap_{t\in T}\mathfrak{G}_t := (\bigcap_{t\in T}V_t, \bigcap_{t\in T}E_t, \bigcap_{t\in T}\nu_t, \bigcap_{t\in T}\kappa_t, \bigcap_{t\in T}\rho_t),$$

respectively. In Fig.1, the concept graph (3) is obviously the union of the concept graphs (1) and (2).

Lemma 1. *The union and intersection of subgraphs of a concept graph $\mathfrak{G}$ is always a subgraph of $\mathfrak{G}$ again.*

Proof. Let $\mathfrak{G}_t := (V_t, E_t, \nu_t, \kappa_t, \rho_t)$ $(t \in T)$ be subgraphs of a concept graphs $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$. Obviously, $\bigcup_{t\in T} V_t \subseteq V$, $\bigcup_{t\in T} E_t \subseteq E$, $\bigcup_{t\in T}\nu_t = \nu|_{\bigcup_{t\in T}E_t}$, $\bigcup_{t\in T}\kappa_t = \kappa|_{\bigcup_{t\in T}V_t \cup \bigcup_{t\in T}E_t}$, and $\bigcup_{t\in T}\rho_t = \rho|_{\bigcup_{t\in T}V_t}$. This affirms the claim for the union. Replacing each $\bigcup$ by a $\bigcap$ in the proof for the union yields a proof for the intersection (notice that $\mathfrak{G}_\emptyset := (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ is a concept graph, too). □

A power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ can be considered as a representation of facts which may function as background knowledge for determining the conceptual content of the concept graphs of $\vec{\mathbb{K}}$ (cf. [Wi03]). From the background knowledge coded in $\vec{\mathbb{K}}$, two types of material inferences shall be made formally explicit (cf. [Br94]): Let $k = 0, 1, 2, \ldots$;

1. *object implications:* for $A, C \subseteq G_k$, $\mathbb{K}_k$ satisfies $A \rightarrow C$ if $A^{I_k} \subseteq C^{I_k}$ and,
2. *concept implications:* for $\mathfrak{B}, \mathfrak{D} \subseteq \mathfrak{B}(\mathbb{K}_k)$, $\mathbb{K}_k$ satisfies $\mathfrak{B} \rightarrow \mathfrak{D}$ if $\bigwedge \mathfrak{B} \leq \bigwedge \mathfrak{D}$.

The *formal implications* $A \rightarrow C$ and $\mathfrak{B} \rightarrow \mathfrak{D}$ give rise to a closure system $\mathcal{C}(\mathbb{K}_k)$ on $\mathbb{S}^{imp}(\mathbb{K}_k) := \{(g, \mathfrak{b}) \in G_k \times \mathfrak{B}(\mathbb{K}_k) \mid g \in Ext(\mathfrak{b})\}$ consisting of all subsets Y of $\mathbb{S}^{imp}(\mathbb{K}_k)$ which have the following property:

(P_k) If $A \times \mathfrak{B} \subseteq Y$ and if $\mathbb{K}_k$ satifies $A \rightarrow C$ and $\mathfrak{B} \rightarrow \mathfrak{D}$ then $C \times \mathfrak{D} \subseteq Y$.

Using the described closure system, the conceptual content of concept graphs (understood as formalized judgments) may be defined as follows: For $k = 1, 2, \ldots$, the *k-ary conceptual content* $C_k(\mathfrak{G})$ (also called the *$\mathbb{K}_k$-conceptual content*) of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of a power context family $\vec{\mathbb{K}}$ is defined as the closure of $\{(\vec{g}, \kappa(e)) \mid e \in E^{(k)} \text{ and } \vec{g} \in \rho(e)\}$ with respect to the closure system $\mathcal{C}(\mathbb{K}_k)$; the *0-ary conceptual content* $C_0(\mathfrak{G})$ (also called *$\mathbb{K}_0$-conceptual content*) of $\mathfrak{G}$ is defined as the closure of $\{(g, \kappa(v)) \mid v \in V \text{ and } g \in \rho(v)\} \cup \{(g_i, (G_0, G_0^{I_0})) \mid \exists((g_1, \ldots, g_k), \mathfrak{c}) \in C_k(\mathfrak{G}) \text{ with } g_i \in \{g_1, \ldots, g_k\}\}$ with respect to the closure system $\mathcal{C}(\mathbb{K}_0)$. Then

$$C(\mathfrak{G}) := C_0(\mathfrak{G}) \,\dot{\cup}\, C_1(\mathfrak{G}) \,\dot{\cup}\, C_2(\mathfrak{G}) \,\dot{\cup}\, \ldots$$

is called the ($\vec{\mathbb{K}}$-)*conceptual content* of the concept graph $\mathfrak{G}$.

The conceptual contents give rise to an *information (quasi-)order* $\lesssim$ on the set of all concept graphs of a power context family: A concept graph $\mathfrak{G}_1 := (V_1, E_1, \nu_1, \kappa_1, \rho_1)$ is said to be *less informative (more general)* than a concept graph $\mathfrak{G}_2 := (V_2, E_2, \nu_2, \kappa_2, \rho_2)$ (in symbols: $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$) if $C_k(\mathfrak{G}_1) \subseteq C_k(\mathfrak{G}_2)$ for $k = 0, 1, 2, \ldots$; $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are called *equivalent* (in symbols: $\mathfrak{G}_1 \sim \mathfrak{G}_2$) if $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$ and $\mathfrak{G}_2 \lesssim \mathfrak{G}_1$ (i.e., $C_k(\mathfrak{G}_1) = C_k(\mathfrak{G}_2)$ for $k = 0, 1, 2, \ldots$). The set of all equivalence classes of concept graphs of a power context family $\vec{\mathbb{K}}$ together with the order induced by the quasi-order $\lesssim$ is an ordered set denoted by $\widetilde{\Gamma}(\vec{\mathbb{K}})$. The ordered set $\widetilde{\Gamma}(\vec{\mathbb{K}})$ is even a complete lattice as shown in [Wi02].

3 Existential Concept Graphs

For defining existential concept graphs as done in [Wi02], the construction of free extensions of power context families by given sets X of variables is needed. To introduce such a construction, for a set X of variables, an *X-interpretation* into a set G_0 with $G_0 \cap X = \emptyset$ is defined as a mapping $\chi : G_0 \cup X \to G_0$ with $\chi(g) = g$ for all $g \in G_0$; the set of all X-interpretations into G_0 is denoted by $B(X, G_0)$.

Now, let $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ be a power context family with $\mathbb{K}_k := (G_k, M_k, I_k)$ for $k = 0, 1, 2, \ldots$ and let X be a set of variables with $G_0 \cap X = \emptyset$. Next, it is explained how the formal contexts $\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots$ are "freely" extented by X:

- let $\mathbb{K}_0[X] := (G_0[X], M_0[X], I_0[X])$ with
  $G_0[X] := G_0 \cup X$, $M_0[X] := M_0$, $I_0[X] := I_0 \cup (X \times \{m \in M_0 \mid \{m\}^{I_0} \neq \emptyset\})$,
- let $\mathbb{K}_k[X] := (G_k[X], M_k[X], I_k[X])$ $(k = 1, 2, \ldots)$ with
  $G_k[X] := \{(u_1, \ldots, u_k) \in G_0[X]^k \mid \exists \chi \in B(X, G_0) : (\chi(u_1), \ldots, \chi(u_k)) \in G_k\}$,
  $M_k[X] := M_k$, and
  $(u_1, \ldots, u_k) I_k[X] m :\iff \exists \chi \in B(X, G_0) : (\chi(u_1), \ldots, \chi(u_k)) I_k m$.

Then $\vec{\mathbb{K}}[X] := (\mathbb{K}_0[X], \mathbb{K}_1[X], \mathbb{K}_2[X], \ldots)$ is again a power context family which is called the *free X-extension* of the power context family $\vec{\mathbb{K}}$. In general, power context families of the form $\vec{\mathbb{K}}[X]$ are called *existential power context families.*

Finally, for defining existential concept graphs, the surjective $\bigwedge$-homomorphisms $\pi_k^X : \underline{\mathfrak{B}}(\mathbb{K}_k[X]) \to \underline{\mathfrak{B}}(\mathbb{K}_k)$ $(k = 0, 1, 2, \ldots)$ are needed which are determined by

$$\pi_k^X(A, B) := (A \cap G_k, (A \cap G_k)^{I_k}) \text{ for } (A, B) \in \underline{\mathfrak{B}}(\mathbb{K}_k[X]).$$

An *existential concept graph* of a power context family $\vec{\mathbb{K}}$ is now defined as a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of a free X-extension $\vec{\mathbb{K}}[X]$ for which an X-interpretation χ into G_0 exists such that $\mathfrak{G}^\chi := (V, E, \nu, \kappa^\chi, \rho^\chi)$ with $\kappa^\chi(u) := \pi_k^X(\kappa(u))$ and $\rho^\chi(v) := \chi(\rho(v))$ is a concept graph of $\vec{\mathbb{K}}$; χ is then called an X-interpretation *admissible on* $\mathfrak{G}$. For a fixed variable set X, $\mathfrak{G}$ is more precisely named an existential concept graph of $\vec{\mathbb{K}}$ *over* X. Examples of existential concept graphs are given in Fig.2.

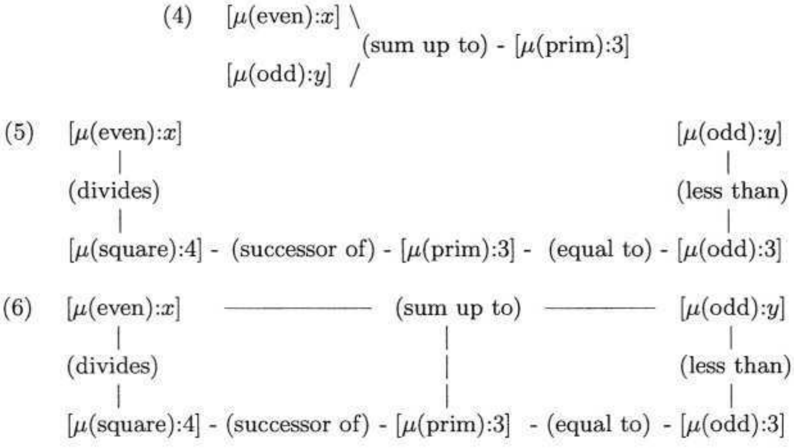

(4)
[μ(even):x] \
(sum up to) - [μ(prim):3]
[μ(odd):y] /
(5)
[μ(even):x]
[μ(odd):y]
(divides)
(less than)
[μ(square):4] - (successor of) - [μ(prim):3] - (equal to) - [μ(odd):3]
(6)
[μ(even):x]
(sum up to)
[μ(odd):y]
(divides)
(less than)
[μ(square):4] - (successor of) - [μ(prim):3] - (equal to) - [μ(odd):3]

Fig. 2. Existential concept graphs of $\vec{\mathbb{K}}(\mathbb{N})$ over the variable set {x,y}

Since an X-interpretation admissible on $\mathfrak{G}$ is also admissible on each subgraph of $\mathfrak{G}$, we immediately get the following lemma:

Lemma 2. *The subgraphs of an existential concept graph over X are existential concept graphs over X, too.*

The *conceptual content* of an existential concept graph $\mathfrak{G}_X$ of a power context family $\vec{\mathbb{K}}$ is defined as the conceptual content of $\mathfrak{G}_X$ understood as a concept graph of the free X-extension $\vec{\mathbb{K}}[X]$. For the understanding of existential concept graphs and their conceptual contents, it is helpful to clarify how variables give rise to object implications of the relational contexts $\mathbb{K}_k[X]$. This is the aim of the following lemma:

Lemma 3. *Let* $\mathbb{K}_k[X] := (G_k[X], M_k[X], I_k[X])$ *with* $k \in \{1, 2, \ldots\}$ *be a relational context of an existential power context family* $\vec{\mathbb{K}}[X]$*; furthermore, let*

α be a map of $G_0 \cup X$ into itself satisfying $\alpha(g) = g$ for all $g \in G_0$. Then $\mathbb{K}_k[X]$ has the object implications $\{(\alpha(u_1),\ldots,\alpha(u_k))\} \longrightarrow \{(u_1,\ldots,u_k)\}$ with $u_1,\ldots,u_k \in G_0 \cup X$.

Proof. Let $((\alpha(u_1),\ldots,\alpha(u_k)),m) \in I_k[X]$. Then there exists an X-interpretation χ into G_0 with $(\chi(\alpha(u_1)),\ldots,\chi(\alpha(u_k)))I_k m$. For the X-interpretation χ_α with $\chi_\alpha(u) := \chi(\alpha(u))$, we obtain $(\chi_\alpha(u_1),\ldots,\chi_\alpha(u_k)) = (\chi(\alpha(u_1)),\ldots,\chi(\alpha(u_k))) \in m^{I_k}$; hence $((u_1,\ldots,u_k),m) \in I_k[X]$. Thus, we proved the desired inclusion $\{(\alpha(u_1),\ldots,\alpha(u_k))\}^{I_k[X]} \subseteq \{(u_1,\ldots,u_k)\}^{I_k[X]}$. □

Let us only consider a special case of applying Lemma 3 to imagine its consequences: For a permutation π of the variable set X, let α_π be the map of $G_0 \cup X$ into itself with $\alpha_\pi(g) = g$ for all $g \in G_0$ and $\alpha_\pi(x) = \pi(x)$ for all $x \in X$. Then we obtain the object implication $\{(\alpha_\pi(u_1),\ldots,\alpha_\pi(u_k))\} \longrightarrow \{(u_1,\ldots,u_k)\}$ with $u_1,\ldots,u_k \in G_0 \cup X$. Together with the corresponding object implication for π^{-1}, this yields that changing variables according to a permutation of X in a (k-ary) object of $\mathbb{K}_k[X]$ does not change the intension of that object.

An existential concept graph $\mathfrak{G}_1 := (V_1,E_1,\nu_1,\kappa_1,\rho_1)$ is said to be *less informative (more general)* than $\mathfrak{G}_2 := (V_2,E_2,\nu_2,\kappa_2,\rho_2)$ (in symbols: $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$) if $C_k(\mathfrak{G}_1) \subseteq C_k(\mathfrak{G}_2)$for $k = 0,1,2,\ldots$; $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are called *equivalent* (in symbols: $\mathfrak{G}_1 \sim \mathfrak{G}_2$) if $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$ and $\mathfrak{G}_2 \lesssim \mathfrak{G}_1$ (i.e., $C_k(\mathfrak{G}_1) = C_k(\mathfrak{G}_2)$ for $k = 0,1,2,\ldots$). The set of all equivalence classes of existential concept graphs of a power context family $\vec{\mathbb{K}}$ over a fixed set X of variables together with the order induced by the quasi-order $\lesssim$ is an ordered set denoted by $\widetilde{\Gamma}(\vec{\mathbb{K}};X)$.

4 Implicational Concept Graphs as Existential Concept Graphs

An *implicational concept graph* of a power context family $\vec{\mathbb{K}}$ is defined as an existential concept graph $\mathfrak{G} := (V,E,\nu,\kappa,\rho)$ of $\vec{\mathbb{K}}$ over a variable set X with a designated pair $(p\mathfrak{G}, c\mathfrak{G})$ of subgraphs such that

1. $\mathfrak{G}$ is the union of $p\mathfrak{G}$ and $c\mathfrak{G}$, and
2. each X-interpretation admissible on $p\mathfrak{G}$ is also admissible on $c\mathfrak{G}$ (and hence on $\mathfrak{G}$ too).

If the implicational nature of $\mathfrak{G}$ shall become more transparent, $p\mathfrak{G} \to c\mathfrak{G}$ may be written instead of $\mathfrak{G}$; the designated subgraphs $p\mathfrak{G}$ and $c\mathfrak{G}$ are called the *premise* and the *conclusion* of the implicational concept graph $\mathfrak{G}$, respectively. An example of an implicational graph is presented by graph (6) in Fig.2 which has graph (4) as premise and graph (5) as conclusion.

To clarify the inferential potential of implicational concept graphs, it is useful to represent implicational concept graphs by attribute implications. How this can be effectively done shall now be explained: For an existential concept graph $\overline{\mathfrak{G}}$ of a power context family $\vec{\mathbb{K}}$ over a variable set X, we introduce the formal context $\mathbb{K}(X;\overline{\mathfrak{G}}) := (B(X,G_0), Sub(\overline{\mathfrak{G}}), \rhd)$ of which

- the object set $B(X, G_0)$ consists of all X-interpretations into the object set G_0 of the formal context $\mathbb{K}_0$ in $\vec{\mathbb{K}}$,
- the attribute set $Sub(\overline{\mathfrak{G}})$ is the set of all subgraphs of $\overline{\mathfrak{G}}$, and
- $\chi \triangleright \mathfrak{G}$ means that the X-interpretation χ is admissible on the subgraph $\mathfrak{G}$ of $\overline{\mathfrak{G}}$.

Proposition 1. $\{\mathfrak{G}_s \mid s \in S\} \to \{\mathfrak{G}_t \mid t \in T\}$ *is an attribute implication of* $\mathbb{K}(X; \overline{\mathfrak{G}})$ *if and only if* $\bigcup_{s \in S} \mathfrak{G}_s \to \bigcup_{t \in T} \mathfrak{G}_t$ *is an implicational concept graph of* $\vec{\mathbb{K}}$ *over* X.

Proof. Let $\{\mathfrak{G}_s \mid s \in S\} \to \{\mathfrak{G}_t \mid t \in T\}$ be an attribute implication of the formal context $\mathbb{K}(X; \overline{\mathfrak{G}})$. By Lemma 1 and 2, $(\bigcup_{s \in S} \mathfrak{G}_s) \cup (\bigcup_{t \in T} \mathfrak{G}_t)$ is an existential concept graph over X having $\bigcup_{s \in S} \mathfrak{G}_s$ and $\bigcup_{t \in T} \mathfrak{G}_t$ as subgraphs. Now, let χ be an X-interpretation admissible on $\bigcup_{s \in S} \mathfrak{G}_s$. Then $\chi \in \mathfrak{G}_s^{\triangleright}$ for all $s \in S$ and hence $\chi \in \mathfrak{G}_t^{\triangleright}$ for all $t \in T$ by our assumed attribute implication; therefore χ is also an X-interpretation admissible on $\bigcup_{t \in S} \mathfrak{G}_t$. Thus, $(\bigcup_{s \in S} \mathfrak{G}_s) \cup (\bigcup_{t \in T} \mathfrak{G}_t)$ is an implicational concept graph of $\vec{\mathbb{K}}$ over X. Conversely, let $\bigcup_{s \in S} \mathfrak{G}_s \to \bigcup_{t \in T} \mathfrak{G}_t$ be an implicational concept graph over X and let $\chi \in \{\mathfrak{G}_s \mid s \in S\}^{\triangleright}$. Since χ is then an X-interpretation admissible on $\bigcup_{s \in S} \mathfrak{G}_s$, χ is also an X-interpretation admissible on $\bigcup_{t \in T} \mathfrak{G}_t$ because of the assumed implicational concept graph. It follows that $\chi \in \{\mathfrak{G}_t \mid t \in T\}^{\triangleright}$ which proves the stated attribute implication. □

Proposition 2. $\mathbb{K}(X; \overline{\mathfrak{G}}) := (B(X, G_0), Sub(\overline{\mathfrak{G}}), \triangleright)$ *is always a formal context of which all extents are non-empty attribute extents. Conversely, let* $\mathbb{K} := (G, M, I)$ *be a clarified formal context of which all extents are non-empty attribute extents; then* $\mathbb{K}$ *is isomorphic to the clarified quotient context of the formal context* $\mathbb{K}(\{x\}; \overline{\mathfrak{G}}) := (B(\{x\}, G), Sub(\overline{\mathfrak{G}}), \triangleright)$ *where* $\overline{\mathfrak{G}} := (V, E, \nu, \kappa, \rho)$ *is the existential concept graph of the power context family* $\vec{\mathbb{K}} := (\mathbb{K})$ *over* $\{x\}$ *with* $V := M$, $E := \emptyset$, $\nu := \emptyset$, $\kappa(m) := \mu m$, *and* $\rho(m) := \{x\}$.

Proof. Since $\mathfrak{G}_\emptyset \in Sub(\overline{\mathfrak{G}})$ and $\mathfrak{G}_\emptyset^{\triangleright} = B(X, G_0)$, each extent of $\mathbb{K}(X; \overline{\mathfrak{G}})$ is non-empty. Let A be any extent of $\mathbb{K}(X; \overline{\mathfrak{G}})$. Then $\mathfrak{G}_A := \bigcup A^{\triangleright} \in Sub(\overline{\mathfrak{G}})$ by Lemma 1. By Proposition 1, we obtain $A^{\triangleright} \to \mathfrak{G}_A$, i.e. $A \subseteq \mathfrak{G}_A^{\triangleright}$. Since each $\mathfrak{G} \in A^{\triangleright}$ is a subgraph of $\mathfrak{G}_A$, it follows that $\mathfrak{G}_A^{\triangleright} \subseteq \mathfrak{G}^{\triangleright}$ for all $\mathfrak{G} \in A^{\triangleright}$ and hence $\mathfrak{G}_A^{\triangleright} \subseteq A^{\triangleright\triangleright} = A$. Thus, $A = \mathfrak{G}_A^{\triangleright}$. Conversely, we consider the clarified quotient context of $\mathbb{K}(\{x\}; \overline{\mathfrak{G}})$ which is the formal context

$$(B(\{x\}, G)/\theta_1, Sub(\overline{\mathfrak{G}})/\theta_2, \triangleright/\theta_3)$$

with $\chi_1 \theta_1 \chi_2 :\iff \chi_1^{\triangleright} = \chi_2^{\triangleright}$, $\mathfrak{G}_1 \theta_2 \mathfrak{G}_2 :\iff \mathfrak{G}_1^{\triangleright} = \mathfrak{G}_2^{\triangleright}$, and $[\chi]\theta_1 (\triangleright/\theta_3) [\mathfrak{G}]\theta_2 :\iff \chi \triangleright \mathfrak{G}$. For $g \in G$, let χ_g be the $\{x\}$-interpretation into G with $\chi_g(x) = g$ and, for $m \in M$, let $\mathfrak{G}_m := (V_m, E_m, \nu_m, \kappa_m, \rho_m)$ be the existential concept graph of $(\mathbb{K})$ with $V_m := m^{II}$, $E_m := \emptyset$, $\nu_m := \emptyset$, $\kappa_m(n) := \mu n$, and $\rho_m(x) := x$. Since $[\chi_g] = [\chi_h] \iff \chi_g^{\triangleright} = \chi_h^{\triangleright} \iff \gamma g = \gamma h \iff g = h$ we have $[\chi_g] = \{\chi_g\}$; furthermore, for $\mathfrak{G}_N \in Sub(\overline{\mathfrak{G}})$ with vertex set N, we have $[\mathfrak{G}_m] = [\mathfrak{G}_N] \iff \mathfrak{G}_m^{\triangleright} = \mathfrak{G}_N^{\triangleright} \iff m^I = N^I \iff \mu m = (N^I, N^{II})$.

Therefore $\alpha : G \to B(\{x\},G)/\theta_1$ with $\alpha(g) := [\chi_g]\theta_1$ is a bijection and $\beta : M \to Sub(\overline{\mathfrak{G}})/\theta_2$ with $\beta(m) := ([\mathfrak{G}_m]\theta_2)$ is a bijection too. Since gIm is equivalent to $[\chi_g]\theta_1(\triangleright/\theta_3)[\mathfrak{G}_m]\theta_2$, it follows that (α,β) is a context isomorphism from $\mathbb{K}$ onto the clarified quotient context of $\mathbb{K}(\{x\};\overline{\mathfrak{G}})$. □

Corollary 1. *The concept lattices of the specific formal contexts* $\mathbb{K}(X;\overline{\mathfrak{G}})$ *are up to isomorphism the concept lattices of formal contexts.*

For further investigations of implicational concept graphs, Proposition 1 and 2 open the rich research area of attribute implications of formal contexts (see for instance Section 2.3 in [GW99a]). Many results on attribute implications carry over so that they have no more to be elaborated and proved in terms of implicational concept graphs. In particular, the important results on bases of attribute implications and their computations and applications are extremely valuable for the theory of implicational concept graphs within Contextual Judgment Logic. Cum grano salis, we could say that the implicational theory of implicational concept graphs is essentially equivalent to the theory of attribute implications.

5 Clausal Concept Graphs as Generalized Implicational Concept Graphs

The Contextual Logic treatment of implications can be extented straightforward to a Contextual Logic treatment of clauses. For this, the definition of an implicational concept graph is generalized as follows: A *clausal concept graph* of a power context family $\vec{\mathbb{K}}$ is defined as an existential concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of $\vec{\mathbb{K}}$ over a variable set X with a designated pair $(p\mathfrak{G}, \{c_t\mathfrak{G} \mid t \in T\})$ consisting of a subgraph $p\mathfrak{G}$ of $\mathfrak{G}$ and a set $\{c_t\mathfrak{G} \mid t \in T\}$ of subgraphs of $\mathfrak{G}$ such that

1. $\mathfrak{G}$ is the union of $p\mathfrak{G}$ and all the $c_t\mathfrak{G}$ with $t \in T$, and
2. each X-interpretation admissible on $p\mathfrak{G}$ is also admissible on at least one $c_t\mathfrak{G}$ with $t \in T$.

If the implicational nature of $\mathfrak{G}$ shall become more transparent, $p\mathfrak{G} \to \bigvee_{t\in T} c_t\mathfrak{G}$ may be written instead of $\mathfrak{G}$; the designated subgraphs $p\mathfrak{G}$ and $c_t\mathfrak{G}$ $(t \in T)$ are called the *premise* and the *disjunctive conclusions* of the clausal concept graph $\mathfrak{G}$, respectively.

As in the case of implicational concepts graphs, the formal contexts $\mathbb{K}(X;\overline{\mathfrak{G}})$ can be used to link clausal concept graphs to Contextual Attribute Logic, the basics of which are outlined in [GW99b] (see also [BS97]). Here, let us only recall the definition of an attribute clause of a formal context $\mathbb{K} := (G, M, I)$: For subsets A and B of the attribute set M, $\bigwedge A \to \bigvee B$ is an *attribute clause* of $\mathbb{K}$ if $g \in A^I$ always implies gIm for at least one $m \in B$. It can be shown that the correspondence between clausal concept graphs and attribute clauses is analogous to the correspondence between implicational concept graphs and attribute implications:

Proposition 3. *Let* $\overline{\mathfrak{G}}$ *be an existential concept graph of a power context family* $\vec{\mathbb{K}}$ *over a variable set* X *and let* $\mathfrak{G}_s$ $(s \in S)$ *and* $\mathfrak{G}_t$ $(t \in T)$ *be subgraphs of* $\overline{\mathfrak{G}}$. *Then* $\bigwedge\{\mathfrak{G}_s \mid s \in S\} \to \bigvee\{\mathfrak{G}_t \mid t \in T\}$ *is an attribute clause of the formal context* $\mathbb{K}(X;\overline{\mathfrak{G}})$ *if and only if* $\bigcup_{s\in S}\mathfrak{G}_s \to \bigvee_{t\in T}\mathfrak{G}_t$ *is a clausal concept graph of* $\vec{\mathbb{K}}$ *over* X.

Proof. Let $\bigwedge\{\mathfrak{G}_s \mid s \in S\} \to \bigvee\{\mathfrak{G}_t \mid t \in T\}$ be an attribute clause of $\mathbb{K}(X;\overline{\mathfrak{G}})$. By Lemma 1 and 2, $(\bigcup_{s\in S}\mathfrak{G}_s) \cup (\bigcup_{t\in T}\mathfrak{G}_t)$ is an existential concept graph over X having $\bigcup_{s\in S}\mathfrak{G}_s$ and the $\mathfrak{G}_t$ $(t \in T)$ as subgraphs. Now, let χ be an X-interpretation admissible on $\bigcup_{s\in S}\mathfrak{G}_s$. Then $\chi \in \mathfrak{G}_s^{\triangleright}$ for all $s \in S$ and hence there is some $t^* \in T$ with $\chi \in \mathfrak{G}_{t^*}^{\triangleright}$ by our assumed attribute clause, i.e., χ is also an X-interpretation admissible on $\mathfrak{G}_{t^*}$. Thus, $(\bigcup_{s\in S}\mathfrak{G}_s) \cup (\bigcup_{t\in T}\mathfrak{G}_t)$ is a clausal concept graph of $\vec{\mathbb{K}}$ over X. Conversely, let $\bigcup_{s\in S}\mathfrak{G}_s \to \bigvee_{t\in T}\mathfrak{G}_t$ be a clausal concept graph over X and let $\chi \in \{\mathfrak{G}_s \mid s \in S\}^{\triangleright}$. Since χ is then an X-interpretation admissible on $\bigcup_{s\in S}\mathfrak{G}_s$, χ is also an X-interpretation admissible on at least one $\mathfrak{G}_{t^*}$ with $t^* \in T$ because of the assumed clausal concept graph. It follows that $\chi \in \mathfrak{G}_{t^*}^{\triangleright}$ for some $t^* \in T$ which proves the stated attribute clause. □

Proposition 2 and 3 show that the theory of clausal concept graphs is essentially equivalent to the theory of attribute clauses of formal contexts. The advantage of this equivalence is that many results about attribute clauses can be transferred to clausal concept graphs which substantially enriches the research on Contextual Judgment Logic. A detailed elaboration of such an enrichment by clausal concept graphs shall be postponed to a subsequent paper.

Here we conclude in particularly pointing out how the expressibility of Contextual Judgment Logic increases by introducing implicational and clausal concept graphs. For instance, $\mathfrak{G}_\emptyset \to \mathfrak{G}$ is an implicational concept graph of a power context family $\vec{\mathbb{K}}$ over a variable set X if and only if all X-interpretations into G_0 are admissible on $\mathfrak{G}$; this shows that certain universal quantifications are expressible by implicational concept graphs. More generally, $\mathfrak{G}_\emptyset \to \bigvee_{t\in T}\mathfrak{G}_t$ is a clausal concept graph of $\vec{\mathbb{K}}$ over X if and only if for all X-interpretations χ into G_0 there exists a $t_\chi \in T$ such that χ is admissible on $\mathfrak{G}_{t_\chi}$; thus, clausal concept graphs allow to express certain disjunctions. For a full understanding of the expressibility of Contextual Judgment Logic, it would especially be necessary to clarify the connections between the logic of implicational and clausal concept graphs and the informationally (quasi-)ordered set of existential concept graphs of power context families; this is a main task for further research.

Acknowledgement. This paper has substantially benefited from the financial support given by the Deutsche Forschungsgemeinschaft which allowed the author to work at the Center for the Study of Language and Information (CSLI) at the Stanford University for five weeks in 2003.

References

[BS97] J. Barwise, L. Seligman: *Information flow: the logic of distributive systems.* Cambridge University Press, Cambridge (UK) 1997.

[Br94] R. B. Brandom: *Making it explicit. Reasoning, representing, and discursive commitment.* Havard University Press, Cambridge 1994.

[GW99a] B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations.* Springer, Heidelberg 1999; German version: Springer, Heidelberg 1996.

[GW99b] B. Ganter, R. Wille: Contextual attribute logic. In: W. Tepfenhart, W. Cyre (eds.): *Conceptual structures: standards and practices.* LNAI **1640**. Springer, Heidelberg 1999, 377–388.

[So84] J. F. Sowa: *Conceptual structures: information processing in mind and machine.* Adison-Wesley, Reading 1984.

[Wi82] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets.* Reidel, Dordrecht-Boston 1982, 445–470.

[Wi00] R. Wille: Contextual Logic summary. In: G. Stumme (ed.): *Working with conceptual structures: Contributions to ICCS 2000.* Shaker-Verlag, Aachen 2000, 265–276.

[Wi02] R. Wille: Existential concept graphs of power context families. In: U. Priss, D. Corbett, G. Angelova (eds.): *Conceptual structures: integration and interfaces.* LNAI **2393**. Springer, Heidelberg 2002, 382–395.

[Wi03] R. Wille: Conceptual content as information - basics for Conceptual Judgment Logic. In: A. de Moor, W. Lex, B. Ganter (eds.): *Conceptual Structures for Knowledge Creation and Communication.* LNAI **2746**. Springer, Heidelberg 2003, 1–15.

[Wi04] R. Wille: A Contextual Logic mathematization of Brandom's inferential semantics (in preparation)

Types and Tokens for Logic with Diagrams

Frithjof Dau

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt
dau@mathematik.tu-darmstadt.de

Abstract. It is well accepted that diagrams play a crucial role in human reasoning. But in mathematics, diagrams are most often only used for visualizations, but it is doubted that diagrams are rigor enough to play an essential role in a proof. This paper takes the opposite point of view: It is argued that rigor formal logic can carried out with diagrams. In order to do that, it is first analyzed which problems can occur in diagrammatic systems, and how a diagrammatic system has to be designed in order to get a rigor logic system. Particularly, it will turn out that a separation between diagrams as representations of structures and these structures themselves is needed, and the structures should be defined mathematically. The argumentation for this point of view will be embedded into a case study, namely the existential graphs of Peirce. In the second part of this paper, the theoretical considerations are practically carried out by providing mathematical definitions for the semantics and the calculus of existential Alpha graphs, and by proving mathematically that the calculus is sound and complete.

1 Motivation and Introduction

The research field of *diagrammatic reasoning* investigates all forms of human reasoning and argumentation wherever diagrams are involved. This research area is constituted from multiple disciplines, including cognitive science and psychology as well as computer science, artificial intelligence, logic and mathematics. But it should not be overlooked that there has been until today a long-standing prejudice against non-symbolic representation in mathematics and logic. Without doubt diagrams are often used in mathematical reasoning, but usually only as illustrations or thought aids. Diagrams, many mathematicians say, are not rigorous enough to be used in a proof, or may even mislead us in a proof. This attitude is captured by the quotation below:

> [The diagram] is only a heuristic to prompt certain trains of inference; ... it is dispensable as a proof-theoretic device; indeed ... it has no proper place in a proof as such. For the proof is a syntactic object consisting only of sentences arranged in a finite and inspectable area.
>
> Neil Tennant 1991, quotation adopted from [Ba93]

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 62–93, 2004.

Nonetheless, there exist some diagrammatic systems which were designed for mathematical reasoning. Well-known examples are Euler circles and Venn diagrams. More important to us, at the dawn of modern logic, two diagrammatic systems had been invented in order to formalize logic. The first system is Frege's Begriffsschrift, where Frege tried to provide a formal universal language. The other one is of more relevance for this conference, as Sowa's conceptual graphs are based on them: It is the systems of existential graphs (EGs) by Charles Sanders Peirce, which he used to study and describe logical argumentation. But none of these systems is used in contemporary mathematical logic. In contrast: For more than a century, linear *symbolic* representation systems (i.e., formal languages which are composed of signs which are a priori meaningless, and which are therefore manipulated by means of purely formal rules) have been the exclusive subject for formal logic. There are only a few logicians who have done research on formal, but non-symbolic logic. The most important ones are without doubt Barwise and Etchemendy. They say that

> there is no principle distinction between inference formalisms that use text and those that use diagrams. One can have rigorous, logically sound (and complete) formal systems based on diagrams.
>
> Barwise and Etchemendy 1994, quotation adopted from [Sh01]

This paper advocates this view that rigor formal logic can carried out by means of manipulating diagrams. The argumentation for this point of view will be embedded into a case study, where the theoretical considerations are practically carried out to formalize the Alpha graphs of Peirce. Although the argumentation is carried out on EGs, it can be transferred to other diagrammatic systems (e.g., for conceptual graphs) as well.

For those readers who are not familiar with EGs, Sec. 2 provides a short introduction into EGs. There are some authors who explored EGs, e.g. Zeman, Roberts or Shin. All these authors treated EGs as graphical entities. In Sec. 3, some of the problems which occur in this handling of EGs are analyzed. For this, the approach of Shin (see [Sh01]) will be used. It will turn out that informal definitions and a missing distinction between EGs and their graphical representations are the main problems. In fact, due to this problems, Shin's (and other authors as well) elaboration of EGs is from a mathematicians point of view insufficient and cannot serve as diagrammatic approach to mathematical logic. I will argue that a separation between EGs as abstract structures and their graphical representations is appropriate, and that a *mathematical* definition for EGs is needed. A question which arises immediately is how the graphical representations should be defined and handled. In Secs. 4 and 6, two approaches to solve this question are presented. It will be shown that a mathematical formalization of the graphical representations may cause a new class of problems. In Sec. 5 we will discuss why mathematical logic does not have to cope with problems which arise in diagrammatic systems. It will turn out that the preciseness of mathematical logic is possible although the separation between formulas and their representation is usually not discussed. From this result we draw the conclusion that a mathematical formalization of the diagrammatic representations of EGs is *not* needed. In Sec. 6, the results of the preceding sections are brought together in order to describe my approach for a mathematical foundation of diagrams. In the remaining sections, the results of the theoretical discussion are applied to elaborate mathematically a complete description of the Alpha-part of EGs.

2 Existential Graphs

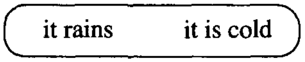
In this paper, we consider the Alpha- and Beta-part of existential graphs. Alpha is a system which corresponds to the propositional calculus of mathematical logic. Beta builds upon Alpha by introducing new symbols to Alpha, and it corresponds to first order predicate logic (FOPL), that is first order logic with predicate names, but without object names and without function names.

We start with the description of Alpha. The EGs of Alpha consist only of predicate names of arity 0, which Peirce called *medads,* and of closed, double-point-free curves which are called *cuts* and used to negate the enclosed subgraph. Medads can be considered as (atomar) propositions, i.e., they correspond to propositional variables in propositional logic. Propositions can be written down on an area (Peirce used the term *'scribing'* instead of 'writing'), and writing down a proposition is to assert it. The area where the proposition is written on is what Peirce called the *sheet of assertion.* It may be a sheet of paper, a blackboard or any other surface. Writing several propositions next to each other (this operation is called a *juxtaposition*) asserts the truth of each proposition, i.e. the juxtaposition corresponds to the conjunction of the juxtaposed propositions. For example, writing the propositions 'it rains' and 'it is cold' next to each other yields the graph

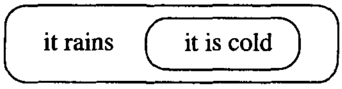
it rains   it is cold

which means 'it rains and it is cold'.

Encircling a proposition is to negate it. The line which is used to encircle a proposition is called a *cut,* the space within a cut is called its *close* or *area.* For example,



has the meaning 'it is not true that it rains and that it is cold', i.e. 'it does not rain or it is not cold'. Cuts may not overlap, but they may be nested. The next graph has two nested cuts.



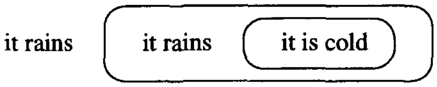
This graph has the meaning ‘it is not true that it rains and that it is not cold’, i.e. ‘if it rains, then it is cold’. The device of two nested cuts is called a *scroll.* From the last example we learn that a scroll can be read as an implication. A scroll with nothing on its first area is called *double cut* (it corresponds to a double negation). As mentioned before, the space within a cut is called the *area* of the cut. In the example above, we therefore have three distinct areas: All the space outside the outer cut, i.e. the sheet of assertion, the space between the outer and the inner cut, which is the area of the outer cut, and the space inside the inner cut, which is the area of the inner cut. An area is *oddly enclosed* if it is enclosed by an odd number of cuts, and it is *evenly enclosed* if it is enclosed by an even number of cuts.

We have the possibility to express conjunction and negation of propositions, thus Alpha has the expressiveness of propositional logic. Peirce also provided a set of five derivation rules for EGs. For Alpha, these rules are:

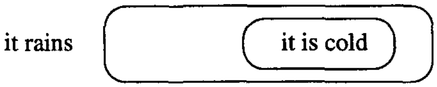
- Erasure: Any evenly enclosed subgraph may be erased.
- Insertion: Any graph may be scribed on any oddly enclosed area.
- Iteration: If a subgraph $\mathfrak{G}$ occurs on the sheet of assertion or in a cut, then a copy of $\mathfrak{G}$ may be scribed on the same or any nested area which does not belong to $\mathfrak{G}$.
- Deiteration: If a subgraph $\mathfrak{G}$ could be the result of iteration, then it may be erased.
- Double Cut: Any double cut may be inserted around or removed from any graph of any area.

This set of rules is sound and complete. In the following, a simple example of a proof is provided (which will be an instantiation of modus ponens in EGs).

Let us start with the graph on the right. It has the meaning ‘it rains, and if it rains, then it is cold’.



The inner instance of ‘it rains’ may be considered a copy of the outer instance of ‘it rains’. Hence we can erase the inner instance of ‘it rains’ using the deiteration-rule.



This graph contains a double cut, which now may be removed.

it rains **it is cold**

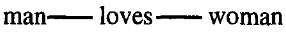
Finally we erase the proposition ‘it rains’ with the erasure-rule.

it is cold

So the graph with the meaning ‘it rains, and if it rains, then it is cold’ implies the graph with the meaning ‘it is cold’.

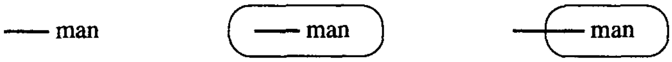
If we go from the part Alpha of EGs to the part Beta, predicate names of arbitrary arity may be used, and a new syntactical item, the *line of identity*

(LI), is introduced. LIs are used to denote both the existence of objects and the identity between objects. They are attached to predicate names and drawn bold. Consider the following graph:



It contains two LIs, hence it denotes two (not necessarily different) objects. The first LI is attached to the unary predicate ‘man’, hence the first object denotes a man. Analogously the second LI denotes a woman. Both lines are attached to the dyadic predicate ‘loves’, i.e. the first object (the man) stands in the relation ‘loves’ to the second object (the woman). The meaning of the graph is therefore ‘there are a man and a woman such that the man loves the woman’, or in short: A man loves a woman.

LIs may cross cuts.[1] Consider the following graphs:



The meaning of the first graph is clear: it is ‘there is a man’. The second graph is built from the first graph by drawing a cut around it, i.e. the first graph is denied. Hence the meaning of the second graph is ‘it is not true that there is a man’, i.e. ‘there is no man’. In the third graph, the LI begins on the sheet of assertion. Hence the existence of the object is asserted and not denied. For this reason the meaning of the third graph is ‘there is something which is not a man’.

Now we have the possibility to express existential quantification, predicates of arbitrary arities, conjunction and negation. Hence we see that the part Beta of EGs corresponds to FOPL.

Essentially, the rules of the calculus for Beta are extensions of the five rules for Alpha such that they now encompass the properties of the lines of identity. For example, the erasure-rule and the insertion-rule are now formulated as follows:

- Erasure: Any evenly enclosed graph and any evenly enclosed portion of a LI may be erased.
- Insertion: Any graph may be scribed on any oddly enclosed area, and two LIs (or portions of lines) oddly enclosed on the same area, may be joined.

[1] This is not fully correct: Peirce denies that a LI may cross a cut (a corollary in [Pe03] states ‘It follows that no line of identity can cross a cut.’), Roberts allows it, and it is not clear whether Shin allows it or not. In Peirce’s view, the third graph has *two* lines of identity which ‘meet’ at the cut. But the discussion of this needs a much deeper understanding of EGs and shall therefore be omitted here.

For the formulation of the remaining three rules we refer to [Ro73] (we have taken the other formulations of the rules from this source).

3 Problems with Existential Graphs

To illustrate some of the problems which may occur in the handling of diagrams, we focus on the EGs as they are described in the book of Shin ([Sh01]), but we will refer to other authors like Peirce, Zeman or Roberts as well. For our discussion, it is sufficient to consider the definitions Shin provides for Beta graphs. It is labelled 'Non-Math, (ematical) Definition' to distinguish it from mathematical definitions as they will appear later in this paper.

Non-math. Definition 1 (Beta Graphs)
The set of beta graphs, $\mathcal{G}_\beta$, is the smallest set satisfying the following:

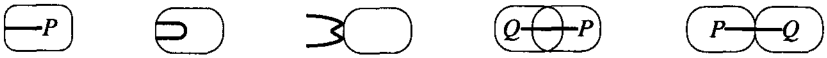
1. *An empty space is in $\mathcal{G}_\beta$.*
2. *A* line of identity *is in $\mathcal{G}_\beta$.*
3. Juxtaposition closure *If G_1 is in $\mathcal{G}_\beta$, …, and G_n is in $\mathcal{G}_\beta$, then the juxtaposition of these n graphs, i.e. $G_1, \ldots, G_n$, (we write '$G_1 \ldots G_n$' for juxtaposition) is also in $\mathcal{G}_\beta$.*
4. Predicate closure *If G is in $\mathcal{G}_\beta$, then a graph with an n-ary* predicate *symbol written at the joint of n loose ends in G is also in $\mathcal{G}_\beta$.*
5. Cut closure *If G is in $\mathcal{G}_\beta$, then a graph in which a* single cut *is drawn in any subpart of G without crossing a predicate symbol is also in $\mathcal{G}_\beta$.*
6. Branch closure *If G is in $\mathcal{G}_\beta$, then a graph in which a line of identity in G* branches *is also in $\mathcal{G}_\beta$.*

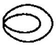
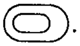
There are two points remarkable in this definition:

1. Although some mathematical terms are used, the definitions are formulated more or less in common spoken language and cannot be seen as *mathematical* definitions.
2. EGs are considered to be *graphical* entities (this can particularly seen in the cut closure rule for Beta graphs).

This approach, particularly the two points mentioned above, yields different kinds of problems which shall be elaborated in this section. We start with problems caused by the use of common language.

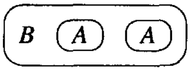
First of all, many important technical terms are defined either in an insufficient way or not at all. For example, the terms *sentence symbol, juxtaposition* or *single cut* (this holds already for Shin's definition of Alpha graphs) as well as the terms *lines of identity, loose ends* or *branching of LIs* are not defined. Even if we have some pre-knowledge on terms like 'single cut' (e.g. we know that a single cut is a closed, double-point-free curve on the plane) or LIs, these definitions leave some issues open. E.g., is it not clear whether two cuts may touch, cross, intersect, or partly overlap, or whether a LI may terminate on a cut. For example, we might ask which of the following diagrams are well-defined diagrams of Beta graphs:

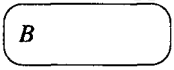
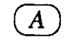


Examining the first three examples, in Shin's book we do not find any example of an Beta graph where a LI terminates on a cut. But, in contrast, this case is explicitly investigated in Peirce's manuscripts or in the book of Roberts. The fourth example is not explicitly excluded by Shin, but it is implicitly excluded based on the background a reader should have of EGs. Considering the fifth example, Peirce used to draw to nested cuts, i.e., scrolls, as follows: . So it seems that a touching of cuts is allowed. But we can raise the question whether scrolls should be handled as own syntactical devices, or whether Peirce's drawing is a sloppy version of two nested cuts which do not touch, i.e. of . So we have to make a *decision* whether cuts may touch or not.

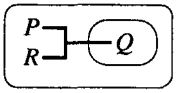
Answering these question is not only about deciding whether a diagram is an EG or not. It is even more important to have rigor definitions for all technical terms when transformation rules, e.g. the rules of a calculus, are applied to EGs. An important example for this is the in most publications undefined term *subgraph*. A rule in the calculus allows us to scribe a copy of a subgraph in the same cut (this is a special case of the *iteration-rule*), which occurs in the treatises of Peirce, Shin, Roberts, and later on in this paper.

To get an impression of the problems, we raise some simple questions on subgraphs. Consider the Alpha graph on the right:

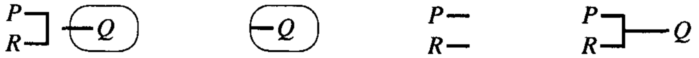
$\mathfrak{G}_\alpha :=$ 

In order to know how the iteration-rule can be applied, it must be possible to answer the following questions: Is  a subgraph of $\mathfrak{G}_\alpha$? Is A A a subgraph of $\mathfrak{G}_\alpha$? Do we have one or two subgraphs  of $\mathfrak{G}_\alpha$ (this is a question which corresponds to the distinction between subformulas and subformula instances in FOPL)?

For Beta, it is important to know how LIs are handled in subgraphs. Consider the following Beta graph:

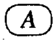
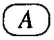
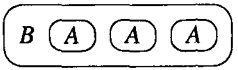
$\mathfrak{G}_\beta :=$ 

We might ask which of the following diagrams are subgraphs of $\mathfrak{G}_\beta$:



Shin (and other authors) does not offer a definition for subgraphs: The answer of the questions above is left to the intuition of the reader. From the discussion above, we draw a first conclusion:

Thesis 1: The definitions of Shin (and other authors) are insufficient for a precise understanding and handling of existential graphs.

If EGs are considered as *graphical* entities, a new class of difficulties has to be coped with. Consider again $\mathfrak{G}_\alpha$. Remember that the iteration-rule should allow us to draw a copy of the subgraph  into the outer cut. But if we want to understand EGs and subgraphs as (graphical) diagrams, then this is obviously *not* possible, because in the outer cut, there is simply not enough space left for another copy of . But, of course, this is not intended by the rule, and everybody who is familiar with EGs will agree that  is a result of a valid application of the iteration-rule. Why that? The idea behind is that we may change the shape of LIs or cuts to a 'certain degree' without changing the meaning of an EG. For this reason it is evident that any attempt which tries to define EGs as purely graphical entities runs into problems.

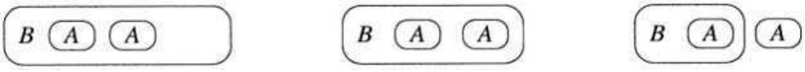
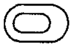


Fig. 1. Diagrams of Alpha graphs

For a discussion of the term 'certain degree', consider the three Alpha graphs in Figure 1. From the left to the right, we decrease the size of the outer cut. Thus there are obviously visual differences between these three diagrams. The question is whether the differences between the first two diagrams are comparable to the differences between the last two diagrams. We have already seen that the shape of a cut is – in some sense – of no relevance. The only thing we have to know is which other items of a graph are enclosed by the cut and which are not. Thus we see that the first two diagrams are (in some sense) the same graph, particularly they have the same meaning. In contrast to that the third diagram has a different meaning and has therefore to be treated differently.

If we – due to the visual differences – treat the first two diagrams to be syntactically different, we would get a syntax which is much too fine-grained. Any kind of equivalence between graphs would be postponed to the semantical level. Furthermore, we would need transformation rules which allow to transform the first graph into the second graph (and vice versa). This syntax would become very complicated and nearly unusable. Thus we see that any appropriate syntax should not distinguish between the first two diagrams.

Now the question arises which properties of the first two diagrams cause us to identify them. Should we syntactically identify graphs when they have the same meaning? This would inappropriately mix up syntax and semantics. For example, the empty sheet of assertion and the graph  have the same meaning, but they should obviously be syntactically distinguished.

In defining a reasonable syntax for the graphs, we see that we have to prescind from certain graphical properties of the diagrams (e.g. the *form* of a cut), while other properties are important (e.g. the number of cuts and other items, or whether an item of the diagram is enclosed by a cut or not). Particularly, EGs should not be understood as graphical entities at all. Instead of this, we have to distinguish between graphs and the diagrams which *represent* the graphs. This is according to Peirce's view. He says: 'A graph [...] is a symbol, and, as such, general, and is accordingly to be distinguished from a graph-replica.' Thus, Peirce distinguishes between *graphs,* which are so-to-speak abstract structures, and their *representations.* Due to this understanding, the first two diagrams in Figure 1 are not different graphs with the same meaning, but different representations, i.e., diagrams, of the same graph, and the third diagram is a representation of a different graph. Peirce explicitly said that arbitrary features of the diagrams may vary, as long as they represent the same diagram. At the beginning of [Pe03] he says:

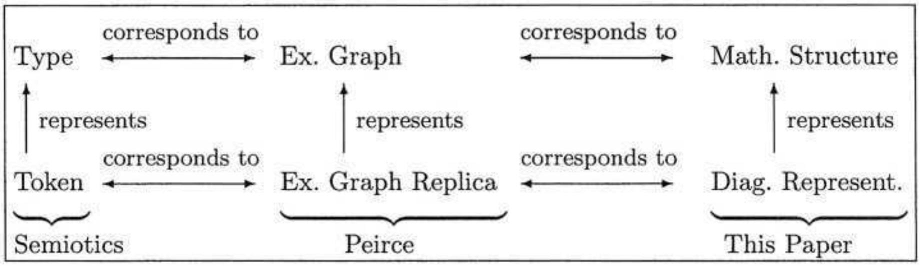
> Convention No. Zero. Any feature of these diagrams that is not expressly or by previous conventions of languages required by the conventions to have a given character may be varied at will. This "convention" is numbered zero, because it is understood in all agreements.

For LIs, he says even more explicit in [Pe03] that 'its shape and length are matters of indifference.'

The distinction between graphs and graph-replicas obviously corresponds to the distinction between *types* (graphs) and *tokens* (graph replicas), as it is known from philosophy. The type-token issue if far from being settled; nonetheless, this important distinction helps us to draw our next conclusion:

> **Thesis 2:** EGs should not be defined as graphical entities. Instead of that, we need a definition of EGs which copes exactly the crucial features of EGs, and the diagrams should be understood as (mere) representations of an underlying EG.

Roughly sketched, we now have the following situation:



4 The First Approach to Diagrams

One of the most important features of mathematics is its preciseness. The preciseness of mathematics is based on a very strict understanding of mathematical definitions and proofs. We have seen that the informal definitions of EGs lead to problems in their understanding and handling. I claim that mathematics is the best language to cope with problems of these kind, i.e.:

> **Thesis 3:** Mathematics provides the highest level of precision available for definitions and proofs.

Thus, in my view, mathematics turns out to be the best instrument for coping the problems discussed in Sec. 3. Particularly, EGs should be defined as mathematical structures (and these structures prescind certain graphical features from diagrams), and the diagrams are representations for these structures. Nonetheless, the last thesis raises the question whether the graph replicas, i.e., the diagrams of EGs, should be defined mathematically as well. This approach shall be discussed in this section.

Let us assume we want to define the graphical representations of Alpha or Beta graphs mathematically. Then we would have two different kinds of objects: Mathematical structures which model EGs, and mathematical structures which model the representations of EGs, i.e., the diagrams. Let us call the first structures *type-structures* and the second structures *token-structures.* In finding a definition for the token-structures, we have two fundamental problems to cope with: First to find a definition for the token-structures which encodes the informally given diagrams as best as possible. Secondly, we have to show how the type-structures are represented by the token-structures. It should be possible to show that each token-structure represents uniquely a type-structure, and that each type-structure is represented by at least one token-structure. Let us call these two principal problems *representation problem.*

An obvious approach is to model the lines of an EG (i.e., the cuts and the LIs) as families of curves in the Euclidean plane $\mathbb{R}^2$. For example, we can model each LI by a smooth, double-point-free curve and each cut by a by a smooth, double-point-free and closed curve.[2] Consider the fist two graphs of Figure 1. They are two different tokens of the same type. Diagrams like these shall be called *type-equivalent* (this term is adopted from [HS02]).

If we define type-equivalence, we have to refer to the relationship between types and tokens. But the diagrams can be compared directly as well. If we consider

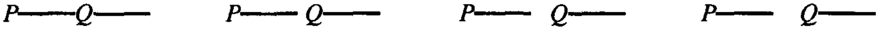
[2] It should be noted that Peirce's understanding of EGs depends on his understanding of the continuum, and this understanding is very different from the set $\mathbb{R}$. Nevertheless a mathematization of the diagrams as a structure of lines and curves in $\mathbb{R}^2$ is convenient as $\mathbb{R}^2$ is the standard mathematization of the Euclidean plane in contemporary mathematics.

again the first two graphs of Figure 1, we see that we have mappings from the cuts resp. the occurences of propositional variables from the first graph to the second which fulfill certain conditions. For example, the mappings are bijective and the preserve some entailment-relations (e.g. if an occurence of a propositional variable or a cut is enclosed by a cut, then this holds for the images as well). In some sense, we can say that the first graph can be topologically transformed into the second graph. Graphs like this shall be called *diagrammatically equivalent* (again, this term is adopted from [HS02]). If we have found adequate definitions for the type- and token-structures as well as for the relation 'a token-structure represents a type-structure', it should be mathematically *provable* that being type-equivalent and being diagrammatically equivalent means the same.

In any description of the diagrams, particularly if we provide a mathematical definition for them, we have to decide some of the bordercases we have discussed in Sec. 3. For example, we have to decide whether (and how often) LIs may touch cuts, or whether cuts may touch or even intersect each other. But in no (reasonable) mathematical definition, we can encode *all* graphical properties of a diagram directly (e.g. by curves). This is easiest to see for letters or words, i.e. the occurences of propositional variables in Alpha graphs or for the relation names in Beta graphs. Of course the *location* of the occurence of a propositional variable in an Alpha graph is important, but neither the size or the font of the propositional variable will be of relevance. Similar considerations hold for relation names in Beta graphs. As the shape of propositional variables or relation names should not be captured by the definition of the diagrams, it is reasonable to handle these items in a different way. The question arises how much of the properties of these items has to captured by a definition of the diagrams. For example, the occurences of propositional variables in an Alpha graph could be modelled by points or spots of the Euclidean plane to which we assign the variables. We see that even in the definition for the diagrams, we have to prescind certain graphical features from diagrams.

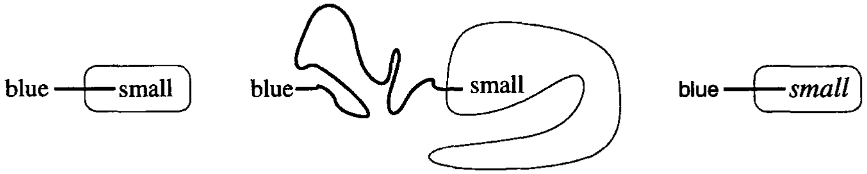
On the other hand: If we try to capture graphical properties of diagrams, some of the items of a diagram will probably be overspecified. For example, how should a simple line of identity – i.e., ——— – be modelled mathematically? We already said that LI could be modelled by a curve in the Euclidean plane. But when a diagram is given, it is usually drawn without providing a coordinate system. Thus, which length should this curve have? Where in the Euclidean plane is it located? Again we see that we cannot capture a diagram exactly by a mathematical definition.

Finally, it is worth to note that we have a couple of (unconscious) heuristics in drawing diagrams for EGs. A simple example is that pending edges should be drawn 'far away enough' from any relation-signs. To see this, consider the following diagrams:



The leftmost diagram should be understood as ‘There is thing which is *P* which is in relation *Q* to another thing’, The rightmost diagram should be understood as ‘There is thing which is *P* and there is thing which is *Q*’. But the meaning of the diagrams in the middle is not clear. So, one will avoid drawing diagrams like these.

Another heuristic is to draw a diagram as simple as possible. Furthermore, although we have seen that neither the size or the font of the propositional variable or a relation name will be of relevance, it is clear that the choice of a font and its size is not arbitrary if a diagram is drawn in a convenient way. This should became clear with the following three diagrams:



Although all diagrams are representations of the same graph (with the meaning ‘there exists something which is blue, but not small), it is clear that the leftmost diagram is the best representation of these three diagrams. Conventions like these cannot by captured by any mathematical definition.

Remember that the reason for finding mathematical definitions for diagrams was to solve the representation problem. We wanted to grasp the distinction between non well-formed and well-formed diagrams, as well as the relationship between graphs and their diagrams, as best as possible. We have already seen that we cannot capture all graphical features of a diagram by a mathematical definition (the more graphical properties are encompassed by a definition, the more technical overhead has to be expected). Now the final question is how a mathematically defined diagram is related to a concrete drawing of a diagram on a sheet of paper. This is a crucial last step from mathematical objects to objects of the real world. Thus, even if we provide mathematical definitions for the diagrams, we still have a representation problem. The initial representation problem between mathematically defined graphs and mathematically defined diagrams has shifted to a representation problem between mathematically defined diagrams and diagrams – i.e., drawings – in the real world. A mathematical definition for diagrams can clarify a lot of ambiguities, but it cannot solve the representation problem finally.

5 Linear Representations of First Order Predicate Logic

In the last section we have raised some questions concerning the representation problem, and we have seen that mathematics alone is not enough to solve these problems. It is likely the often unclear relationship between diagrams and represented structures which causes mathematicians to believe that diagrams cannot have a proper place in mathematical argumentation, esp. proofs. It is argued that only a symbolic system for logic can provide the preciseness which is needed in mathematical proofs (for a broader discussion of this see [Sh01]). It seems that the problems we have discussed in the last section simply do not occur in mathematics, esp. mathematical logic. In this section we will have a closer look on this. We start with a definition of the well-formed formulas of FOPL.

Definition 2. *The alphabet for first order logic consists of the following signs:*

- *Variables:* $x_1, x_2, x_3, \ldots$ *(countably many)*
- *Relation symbols:* $R_1, R_2, R_3, \ldots$ *(countably many). To each predicate symbol* R_i *we assign an* arity $ar(R_i) \in \mathbb{N}$.
- *Constant symbols:* $c_1, c_2, c_3, \ldots\ldots$ *(countably many)*
- *Connectives:* $\wedge, \neg, \exists$
- *Auxiliary Symbols:* .,,,(,)

Definition 3. *The formulas of FOPL are inductively defined as follows:*

1. *Each variable and each constant name is a* term.
2. *If R is a predicate symbol with arity* n *and if* $t_1, \ldots, t_n$ *are terms, then* $f := R(t_1, \ldots, t_n)$ *is a formula.*
3. *If* f' *is a formula, then* $f := \neg f'$ *is a formula.*
4. *If* f_1 *and* f_2 *are formulas, then* $f := (f_1 \wedge f_2)$ *is a formula.*
5. *If* f' *is a formula and* α *is a variable, then* $:= \exists \alpha . f'$ *is a formula.*

It is easy to capture the idea behind this definitions: First, we fix a set of *signs,* and a formula is a *sequence* of these signs which has been composed according to certain *rules.*

Let us consider the following two strings (the relation name R_1 has arity 2):

$$\exists x_1 . \exists x_2 . R_1(x_1, x_2)$$
$$\exists x_1 .\ \exists x_2 .\ R_1(x_1, x_2)$$

Although these two strings are written in different places and although they look slightly different, they clearly represent the same formula. For our considerations, it is worth to raise the question which steps a reader has to pass from the perception of a string to the identification of a formula which is represented by the string. Roughly spoken, if a reader reads the two strings above, she passes the following steps:

1. The region on the paper (the blackboard, ...) must be identified where the representation of the formula is written on. In the example above, this is possible because we have written the two different strings into two different lines.

2. In this region, we must be able to identify representations of the *signs* which may occur in a formula (e.g. the sign '$\exists$', which appears twice, or the sign 'R_1', which appears only once). Nothing else may occur.
3. The representations of the signs must be assembled in a way such that we are able to identify their ordering on the region. That is: We must be able to identify the *sequence.* For our examples, we identify the ordering of the instances of signs by reading them from left to right.
4. Finally, after we have reconstructed the sequence of signs (internally), we can check whether this sequence is a well-defined formula, i.e., whether it is composed with the rules of Definition 3.

In the following, we will use the label $(*)$ to refer to these four steps. The process $(*)$ yields the same result for the two strings above: In both cases, the lines represent the same sequence of signs, which is in fact a well-formed formula. Thus every mathematician would (hopefully) agree that these two strings represent the same (well-defined) formula.

We want to stress that the process of perceiving a representation of a formula is not 'deterministic': It is not clear without ambiguity for each string whether it represents a formula or not. To see this, consider the following strings (the 'type-setting problems' are intended). The question is: Which of these strings represents a well-defined formula?

$$\exists\!\!\!x_1\!\!\exists x_2\!\!R_1\!\!(\!x_1\!,\!x_2) \tag{1}$$

$$\begin{array}{llll} \exists & .^{x_2}\, R_1(& &) \\ x_1 & \exists & \dot{:} & x_1{}'x_2 \end{array} \tag{2}$$

$$\exists x_1.\exists\heartsuit.R_1(x_1,x_2) \tag{3}$$

$$),\exists..R_1x_1\exists x_1(x_2x_2 \tag{4}$$

$$\exists \mathrm{x}_1\,.\exists \mathrm{x}_2\,.R_1(\mathrm{x}_1,\mathrm{x}_2) \tag{5}$$

$$\scriptstyle \exists x_1.\exists x_2.R_1(x_1,x_2) \tag{6}$$

$$\exists x_1.\exists_{x_2.R_1}(x_1,x_2) \tag{7}$$

$$\exists x_1. \qquad \exists x_2\,.R_1(\qquad x_1,x_2) \tag{8}$$

$$\scriptstyle\exists \qquad \textstyle x_1. \qquad \exists_{x_2.R_1}(\qquad x_1,x_2) \tag{9}$$

In line 1, we are neither able to identify the signs, nor to identify their ordering. That is we cannot pass the steps (2) and (3) of $(*)$, thus this line does not represent a formula. In line 2, we are able to identify the signs, but we are not able to identify their ordering. That is we cannot pass the step (3) of $(*)$, thus this line does not represent a formula as well. Moreover, it may be doubted whether step 1 of $(*)$ can be passed without problems. In line 3, we are able to identify the signs, but one of these signs does obviously not belong to our alphabet of first order logic, thus this line does not represent a formula. In line 4, we are able to identify the signs, all of them belong to to our alphabet of first order logic, and we are able to identify their ordering, that is we reconstruct a sequence of signs of our alphabet. But we see that this sequence is not build according to our rules (we cannot pass the step (4) of $(*)$.). Thus this line does not represent a formula. In the remaining lines, it is not uniquely determined whether the lines represent a formula or not. In line 5, the font for the variables has changed. In mathematical texts, different fonts are often used to denote mathematical entities of different kinds, but this is not a general rule. So it depends on the context whether lines 5 or 6 are accepted to represent formulas. Using different sizes of a font is usually driven by a specific purpose. The same holds for the use of significantly different distances between signs. It is hardly conceivable to find a purpose for using different font sizes or significantly different distances in formulas. Thus it is possible, but not sure, that the lines 7–9 are not accepted by a mathematician to represent a formula.

In standard books on logic, usually only the last step of $(*)$ is discussed. The main reason for this is the following: The linear notion of formulas corresponds the way ordinary text is written: Text is assembled of letters which are written side by side and which are read from left to right. As we are used to read texts, we are trained as well to read strings which shall represent formulas. Thus the first three steps of $(*)$ are unconsciously executed when we perceive a representation of a formula.

But as soon we perceive an unfamiliar representation (like in the strings 1-9), we become aware of the whole process described by $(*)$. We realize that mathematical structures need representations, and in mathematics we have a clear separation between structures and their representations. The representations rely on conventions, either implicit or explicit, based on common cultural background as well as on mathematical socialization, and they are not fully explicated. Nonetheless, this usually poses no problems: Although these conventions can never be fully explicated, as long as we provide representations of mathematical structures in accordance to these conventions, they are strong enough to provide a secure transformation from the external representation of a structure (e.g. on a sheet of paper or on a blackboard) into an internal representation of any mathematician, i.e., they refer to the represented structures in a clear and and non-ambiguous way (as Barwise says in a broader discussion of representations: ‘Every representation indicates a genuine possibility’ [Ba93]).

The next thesis claims that this approach can be adopted for diagrammatic systems.

Thesis 4: In a mathematical theory, the mathematical structures need representations. A rigor mathematical theory can be developed without providing mathematical definitions for the representations. Instead of that, it is sufficient if we have conventions – either implicit or explicit — which describe the representations, as well as the relationship between structures and their representations, in a clear and non-ambiguous way.

6 The Second Approach to Diagrams

In Sec. 3, we have argued that in literature, EGs are described in an informal and insufficient way (Thesis 1). Furthermore, we should not mix up graphs and their representations. Particularly, EGs should be defined as formal structures and not as graphical entities (Thesis 2). Thesis 3 claims that mathematics is the best method to describe formal structures. From this we can conclude that EGs should be defined as *mathematical* structures. Nonetheless, we haven't already solved the question whether the representations should be defined mathematically as well.

In Sec. 4, we presented some difficulties when we try to define the diagrams mathematically. The arguments of Sec. 4 are not strong enough to claim that a mathematical definition of diagrams will always run into problems. In contrast: Finding a appropriate mathematical definition for the diagrams should clarify the points mentioned in Sec. 3. That is a mathematical definition would make clear without ambiguities which diagrams should be considered to be well-formed diagrams of EGs, and which not. Furthermore, the relation between graphs and their representations can be elaborated mathematically as well. But providing mathematical definitions for the diagrams may result in a technical overhead or overspecification of the formalization of EGs and their representations, and none mathematical definition can solve the representation problem finally.

On the other hand, as seen in Sec. 5, mathematical logic is a mathematical theory, although the representations of the types in logic, i.e., formulas, are not fully explicated, but they rely on different conventions.

In contrast to the attempt to capture the representations by mathematical definitions, we have seen that logic is a rigor mathematical theory, although representations are only captured not fully explicated conventions. This works because these conventions are mainly based on a common and solid cultural background, namely the form how text is presented. For diagrams, we have a lack of conventions.

Thus, in both approaches, we have to *provide* a set of conventions on how diagrams are written. Particularly for EGs, we have to make clear without ambiguities which diagrams shall be considered to be well-formed diagrams of EGs. In the first approach, these conventions are provided informally in common language. In the second approach, these conventions are described by a mathematical definition. Thus, the mathematical definition should capture as best as possible the informally provided conventions of the first approach. The advantage of a mathematical definition is its preciseness, but we gain this preciseness for the cost of a technical overspecification of the diagrams. Furthermore, we have seen that a mathematical definition cannot capture exactly the diagrams.

As already said above, arguments like these are not strong enough to generally discard mathematical definitions for the token-structures. It has to be estimated whether a mathematical definition is really needed, or whether the conventions for drawing diagrams and for the relationship between representing diagrams and represented structures can be captured sufficiently by descriptions in common language. If the latter is possible, we are able to gain the rigorousness and preciseness of a mathematical theory without a technical overspecification of the diagrams. Thus, in this case, I claim that this approach should be preferred.

We have already mentioned that the Alpha system of EG is said to be equivalent to propositional logic, the Beta system of EG is said to be equivalent to first order predicate logic. In fact, we find good arguments for that in the books of Zeman, Roberts or Shin. But, as EGs are described informal and insufficiently, these argumentation cannot be seen as (mathematical) proofs. With our approach, we can solve these problems: If we define EGs as mathematical structures, we are now able to *prove mathematically* the equivalences between the Alpha system and the Beta system to propositional and first order predicate logic, respectively.

In the following section, we exemplify this approach for Alpha graphs. Of course this is a fairly simple example. Its advantage is that we can work out the definitions of Alpha and the proof of the equivalence to propositional logic on a couple of pages. Its disadvantage is that the problems we discussed in Sec. 3 appear much stronger in Beta, thus the benefit of a mathematical foundation of graphs cannot be seen that clearly for Alpha. An elaboration of this approach for the Beta system is in progress (a first step towards Beta can be found in [Da03]).

7 Syntax for Alpha Graphs

Alpha graphs are built up from two syntactical devices: sentence symbols and cuts. We first fix the sentence symbols, which we call *propositional variables.*

Definition 4 (Propositional Variables).
Let $\mathcal{P} := \{P_1, P_2, P_3, \ldots\}$ *be a countably infinite set of* propositional variables.

Alpha graphs can be seen to built up from propositional variables and cuts by an appropriate inductive definition. We can try to transform this into a inductive mathematical definition. This is possible, but here Alpha graphs are defined in one step. To emphasize that these structures are abstract structures, not diagrams, they are termed *formal* Alpha graphs.

Let us briefly discuss what we have to capture in the mathematical definition. Consider first the two diagrams depicted in Figure 2.

According to the discussion in Secs. 3–6, these two diagrams represent the same Alpha graph: In fact, we have a one-to-one-correspondence between the cuts resp. the occurences of propositional variables in both diagrams such that a cut

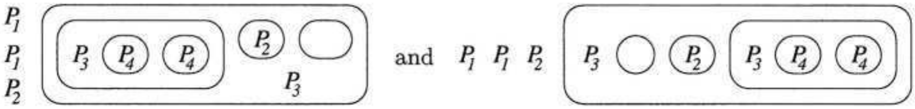


Fig. 2. Two Diagrams of the same Alpha graph

or an occurence of a propositional variable is enclosed by another cut in the first diagram if and only if this holds for their counterpart in the second diagram. For example, in both diagrams of Figure 2 the outermost cut encloses exactly all other cuts, one occurence of the propositional variable P_2 and all occurences of the propositional variables P_3 and P_4. We will say more specifically that the occurence of the propositional variable P_2 and the two cuts are enclosed *directly,* while all other enclosed items are enclosed *indirectly,* and we will say that the items which are directly enclosed by a cut are placed in the *area* of this cut. It is convenient to say that the outermost items (the outermost cut, the two occurences of P_1 and one occurence of P_2 are placed on the area of the sheet of assertion. This enclosing-relation is the main structural relation between cuts and the other items of a graph.

A mathematical definition of Alpha graphs must therefore capture the following:

1. A set of occurences of propositional variables,
2. a set of cuts, and
3. the above-mentioned enclosing-relation.

To distinguish between propositional variables and occurences of propositional variables, we will introduce *vertices* which are labelled with propositional variables. This yields the possibility that a propositional variable may occur several times in a graph, even in the same cut. The cuts are introduced as elements of a set *Cut.* It is reasonable to introduce the sheet of assertion as own syntactical device. The enclosing-relation will be captured by a mapping *area,* which assigns to each cut (and to the sheet of assertion) the set of all other elements of the graph which are directly enclosed by the cut. We know that we have some restrictions for cuts, e.g. cuts may not overlap. These restrictions are captured mathematically by conditions for the mapping *area.* A possible definition for formal Alpha graphs is the following:

Definition 5 (Formal Alpha Graph).

A formal Alpha graph *is a 5-tuple* $(V, \top, Cut, area, \kappa)$ *with*

- *V and Cut are disjoint, finite sets whose elements are called* vertices *and* cuts, *respectively,*
- $\top$ *is a single element with* $\top \notin V \cup Cut$, *called the* sheet of assertion,
- $area : Cut \cup \{\top\} \to \mathfrak{P}(V \cup Cut)$ *is a mapping such that*

a) $c_1 \neq c_2 \Rightarrow area(c_1) \cap area(c_2) = \emptyset$,
b) $V \cup E \cup Cut = \bigcup_{d \in Cut \cup \{\top\}} area(d)$,
c) $c \notin area^n(c)$ *for each* $c \in Cut \cup \{\top\}$ *and* $n \in \mathbb{N}$ *(with* $area^0(c) := \{c\}$ *and* $area^{n+1}(c) := \bigcup\{area(d) \mid d \in area^n(c)\}$*), and*

– $\kappa : V \to \mathcal{P}$ *is a mapping.*

The elements of $Cut \cup \{\top\}$ *are called* contexts. *As we have for every* $x \in V \cup Cut$ *exactly one context* c *with* $x \in area(c)$, *we can write* $c = area^{-1}(x)$ *for every* $x \in area(c)$, *or even more simple and suggestive:* $c = ctx(x)$.

In this definition, we have already captured some important technical terms. Mainly we have defined the *sheet of assertion* and the *cuts* of formal Alpha graphs, and we have introduced the informal described mapping *area.* In the following, we will often speak more simply of 'graphs' instead of 'formal Alpha graphs'.

There is a crucial difference between formal Alpha graphs and most other languages of logic: Usually, the well-formed formulas of a language are built up inductively. In contrast to that, formal Alpha graphs are defined in one step. The structure of a formula in an inductively defined language is given by its inductive construction. Of course we know that Alpha graphs bear a structure as well: A cut of the graph may contain other cuts, but cuts may not intersect. Thus, for two (different) cuts, we have three possibilities: The first cut encloses the second one, the second cut encloses the first one, or the two cuts are incomparable. If we incorporate the sheet of assertion into this consideration, it has to be expected that this idea induces an order $\leq$ on the contexts which should be a tree, having the sheet of assertion $\top$ as greatest element. As we now have a mathematical definition of Alpha graphs, we must be able to *prove* this proposition.[3]

In the next definition, we define an ordering on the vertices and edges which will capture the enclosing-relation.

Definition 6 (Ordering on the Contexts).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *be a graph. We define a relation* $\leq$ *on* $Cut \cup \{\top\}$ *as follows:* $d \leq c \;:\Longleftrightarrow\; d \in \bigcup_{n=0}^{\infty} area^n(c)$.

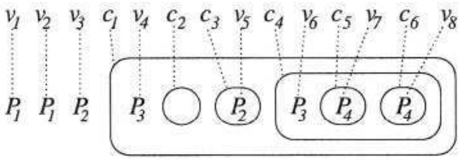
[3] As we mathematize *informal* given entities (here: Alpha graphs), we cannot *prove* (mathematically) that the mathematization, e.g. the definition of formal Alpha graphs, is 'right'. So the attempt to prove that we have an induced tree on the context can be understood as a test on our mathematical 're-engineering' of Alpha graphs. If we cannot prove this proposition, our mathematization of Alpha graphs does not capture crucial features of Alpha graphs, thus we should rework the definition. If we can prove this proposition, this is a good argument that our definition is 'right'.

We set $x < y :\iff x \leq y \wedge y \nleq x$ *and* $x \lneq y :\iff x \leq y \wedge y \neq x$ *to avoid misunderstandings. For* $c \in Cut \cup \{\top\}$, *we set* $\leq[c] := \{x \in V \cup Cut \cup \{\top\} \mid x \leq c\}$ *and* $\lneq[c] := \{x \in V \cup Cut \cup \{\top\} \mid x \lneq c\}$. *Every element* x *of* $\lneq[c]$ *is said to be* enclosed by *c, and vice versa: c is said to* enclose x. *For every element of* $area(c)$, *we say more specifically that it is* directly enclosed by c.

Analogously to [Da03], we get the following lemma:

Lemma 1. *Let* $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *be a graph. Then* $\leq$ *is an order on* $Cut \cup \{\top\}$ *which is a tree with the sheet of assertion* $\top$ *as greatest element.*

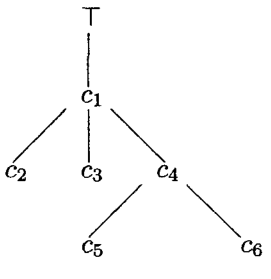
To provide an example of a formal Alpha graph, we consider the right diagram of Figure 2. Additionally, we have labelled the vertices and cuts with names for pairwise distinct elements. Below the diagram, the formal Alpha graph which is represented by the diagram is provided.



$$\begin{array}{ll}
\mathfrak{G} := (\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\},\top,\{c_1,c_2,c_3,c_4,c_5,c_6\}, & \} \; V,\top,Cut \\
\left.\begin{array}{l} \quad\{(\top,\{v_1,v_2,v_3,c_1\}),(c_1,\{v_4,c_2,c_3,c_4\}),(c_2,\emptyset), \\ \quad\;(c_3,\{v_5\}),(c_4,\{v_6,c_5,c_6\}),(c_5,\{v_7\}),(c_6,\{v_8\})\}, \end{array}\right. & \left.\vphantom{\begin{array}{l}a\\a\end{array}}\right\} \; area \\
\left.\begin{array}{l} \quad\{(v_1,P_1),(v_2,P_1),(v_3,P_2),(v_4,P_3), \\ \quad\;(v_5,P_2),(v_6,P_3),(v_7,P_4),(v_8,P_4)\}) \end{array}\right. & \left.\vphantom{\begin{array}{l}a\\a\end{array}}\right\} \; \kappa
\end{array}$$

Next we show the mapping *ctx*, and the quasiorder $\leq$ is presented by its Hasse-diagram.

x	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	c_1	c_2	c_3	c_4	c_5	c_6
$ctx(x)$	$\top$	$\top$	$\top$	c_1	c_3	c_4	c_5	c_6	$\top$	c_1	c_1	c_1	c_4	c_4



Note that we did not further specify the objects $v_1, \ldots, v_8$, $\top$, and $c_1,, \ldots, c_6$. It is quite obvious that a diagram of an EG cannot determine the mathematical objects for vertices or cuts, it only determines the *relationships* between these objects. In fact we can choose arbitrary sets for these mathematical objects (we only have to take into account that $\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\}$, $\{\top\}$,

$\{c_1, c_2, c_3, c_4, c_5, c_6\}$ must be pairwise disjoint). In other words: The diagram of a graph determines the graph only up to *isomorphism.* The isomorphism-relation is canonically defined as follows:

Definition 7 (Isomorphism).

Let $\mathfrak{G}_i := (V_i, \top_i, Cut_i, area_i, \kappa_i)$, $i = 1, 2$ *be two formal Alpha graphs. Then we call* $f = f_V \cup f_{Cut}$ isomorphism, *if* $f_V : V_1 \to V_2$ *and* $f_{Cut} : Cut_1 \cup \{\top_1\} \to Cut_2 \cup \{\top_2\}$ *are bijective with* $f_{Cut}(\top_1) = \top_2$ *such that* $f[area_1(c)] = area_2(f(c))$ *for each* $c \in Cut_1 \cup \{\top_1\}$ *(we write* $f[X]$ *for* $\{f(x) \mid x \in X\}$*), and* $\kappa_1(v) = \kappa_2(f_V(v))$ *for all* $v \in V_1$*.*

From now on, isomorphic graphs are implicitly identified.

We have seen that a *diagram of an Alpha graph* is a diagram which is built up from two different kinds of items, namely of closed, double-point-free and smooth curves which represent cuts (we will call them *cut-lines*), and signs which denote the propositional variables P_i, such that two different items of the diagram neither overlap nor intersect.

Of course each formal Alpha graph can be represented by an appropriate diagram. This diagram can be constructed iteratively over the tree of its context, using Lem. 1. Let on the other hand a diagram be given. It is easy to find (up to isomorphism) the corresponding formal Alpha graph $(V, \top, Cut, area, \kappa)$: We choose sets V and Cut of which its elements shall stand for the occurences of propositional variables resp. the cut-lines in the diagram, and the mapping κ is defined accordingly. Then, the mapping area *area* is now defined as follows: Let $c \in Cut$ be a cut. So we have a uniquely given cut-line *cl* in our diagram which corresponds to c. Now let $area(c)$ be the set of all $x \in V \dot\cup Cut$ such that x corresponds to an item of the diagram (an occurence of a PV or a cut-line) which is directly enclosed by the cut-line *cl.* Furthermore, let $area(\top)$ be the set of all $x \in V \dot\cup Cut$ such that x corresponds to an item which is not enclosed by any cut-line. So we obtain a formal Alpha graph which is represented by the diagram. (The correspondence between formal Alpha graphs and diagrams can be much more explicated, but this is omitted due to space limitation).

Next we will define mathematically what a *subgraph* of a formal Alpha graph is.

Definition 8 (Subgraph).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *be a graph. The graph* $\mathfrak{G}' := (V', \top', Cut', area', \kappa')$ *is called a* subgraph of $\mathfrak{G}$ in the context $\top'$ *if*

- $V' \subseteq V$, $Cut' \subseteq Cut$ *and* $\top' \in Cut \cup \{\top\}$,
- $area'(\top') = area(\top') \cap (V' \cup Cut')$ *and* $area'(d) = area(d)$ *for each* $d \in Cut'$,
- $ctx(x) \in Cut' \cup \{\top'\}$ *for each* $x \in V' \cup Cut'$, *and*
- *the mapping* κ' *is the restriction of* κ *to* V'.

We write: $\mathfrak{G}' \subseteq \mathfrak{G}$ *and* $area^{-1}(\mathfrak{G}') = \top'$ *resp.* $ctx(\mathfrak{G}') = \top'$.

We provide some examples for this definition, based on the graph of Figure 2. We consider substructures of this graphs which are shaded in the diagram.

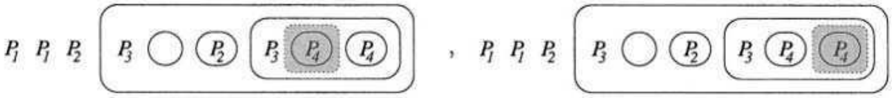


Fig. 3. Different subgraphs

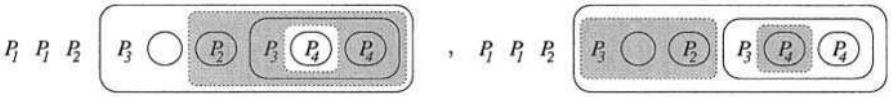


Fig. 4. Two substructures which are no subgraphs

If we consider a graph and a context of this graph, then this context together with all cuts and vertices enclosed by this context form a subgraph. This is a special case for subgraphs which is captured by the following definition and lemma.

Definition 9 (Subgraph induced by a Cut).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *be a formal Alpha graph and* $c \in Cut \cup \{\top\}$ *be a context. The graph* $\mathfrak{G}[c] := (V', \top', Cut', area', \kappa')$ *is defined as follows:* $V' := \leq[c] \cap V$, $Cut' := \leq[c] \cap Cut$, $\top' := c$, $area' := area|_{Cut' \cup \{\top'\}}$, *and* $\kappa' := \kappa|_{V'}$.

Lemma 2. *If* $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *is a formal graph and* $c \in Cut \cup \{\top\}$ *is a context, then* $\mathfrak{G}[c]$ *is a subgraph of* $\mathfrak{G}$.

Proof: Trivial.

The next two definitions are neccessary for defining and understanding the calculus we will present in Sec. 8. Most of the rules in this calculus modify only parts of a graph which are enclosed by a specific context. For some rules we have to distinguish whether this context is enclosed by an odd or even number of cuts. For this reason Definition 10 is needed. As we modify the graph only 'inside' the specific context, the part of the graph outside of this context remains unchanged. We will say that the starting graph and the resulting graph are isomorphic except for the context, which is captured by Definition 11.

Definition 10 (Even and Oddly Enclosed, Pos. and Neg. Contexts).

Let $\mathfrak{G} = (V, \top, Cut, area, \kappa)$ *be a formal Alpha graph, let* x *be a subgraph or let* x *be an element of* $V \cup Cut \cup \{\top\}$. *We set* $n := |\{c \in Cut \mid x \in \leq[c]\}|$. *If* n *is even,* x *is said to be* evenly enclosed, *otherwise* x *is said to be* oddly enclosed.

The sheet of assertion $\top$ *and each oddly enclosed cut is called a* positive context, *and each an evenly enclosed cut is called* negative context.

Definition 11 (Partial Isomorphism).

For $i = 1, 2$, *let* $\mathfrak{G}_i := (V_i, \top_i, Cut_i, area_i, \kappa_i)$, *be two formal Alpha graphs and let* $c_i \in Cut_i \cup \{\top_i\}$ *be given contexts. For* $i = 1, 2$, *we set* $V_i' := \{v \in V_i \mid v \not\leq c_i\}$ *and* $Cut_i{}' := \{d \in Cut_i \cup \{\top_i\} \mid d \not< c_i\}$. *Then* $f = f_V \dot\cup f_{Cut}$ *is called* isomorphism except for $c_1 \in Cut_1 \cup \{\top_1\}$ and $c_2 \in Cut_2 \cup \{\top_2\}$ *if* $f_V : V_1' \to V_2'$ *and* $f_{Cut} : Cut_1{}' \to Cut_2{}'$ *are bijective with* $f_{Cut}(\top_1) = \top_2$, *such that* $f[area(c)] = area'(f(c))$ *for each* $c \in Cut_1 \cup \{\top_1\}'$, *and* $\kappa_1(v) = \kappa_2(f_V(v))$ *for all* $v \in V_1'$.

A common operation on diagrams is to *juxtapose* them, i.e. writing them side by side. On the side of the mathematical structure, this corresponds to the disjoint union of a set of graphs, which is captured by the next definition.

Definition 12 (Juxtaposition of Formal Alpha Graphs).

Let $\mathfrak{G}_i := (V_i, \top_i, Cut_i, area_i, \kappa_i)$ *be graphs for* $i = 1, \ldots, n$ *with* $n \in \mathbb{N}_0$. *The* juxtaposition of the $\mathfrak{G}_i$ *is defined to be the following graph* $\mathfrak{G} := (V, \top, Cut, area, \kappa)$*:*

- $V := \bigcup_{i=1,\ldots,n} V_i \times \{i\}$,
- $Cut := \bigcup_{i=1,\ldots,n} Cut_i \times \{i\}$,
- *area is defined as follows:* $\begin{array}{c} area((c,i)) = area_i(c) \times \{i\} \text{ for } c \in Cut_i \\ area(\top) = \bigcup_{i=1,\ldots,n} area_i(\top_i) \times \{i\} \end{array}$ *, and*
- $\kappa(v, i) := \kappa_i(v)$ *for all* $v \in VE$ *and* $i = 1, \ldots, n$.

In the graphical notation, the juxtaposition of the $\mathfrak{G}_i$ *is simply noted by writing the graphs next to each other, i.e. we write:* $\mathfrak{G}_1\ \mathfrak{G}_2\ \ldots\ \mathfrak{G}_n$.

It should be noted that the juxtaposition of an empty set of graphs is allowed, too. It yields the empty graph, i.e. $(\emptyset, \top, \emptyset, \emptyset, \emptyset)$.

8 Semantics and Calculus for Formal Alpha Graphs

We start this section with providing a semantics for formal Alpha graphs. As in propositional logic, we assign truth values to the propositional variables by *valuations*. Propositional variables stand for propositions which are simply true or false.

Definition 13 (Valuation). *A* valuation *is a mapping* $val : \mathcal{P} \to \{\mathtt{ff}, \mathtt{tt}\}$.

Now we have to extend valuations to graphs by reflection of the meaning of cuts and juxtaposition. This is done close to the so-called *endoporeutic method* of Peirce.

Definition 14 (Evaluations).

Let val be a valuation and let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ *be a graph. We evaluate* $\mathfrak{G}$ *for val inductively over* $c \in Cut \cup \{\top\}$*. The evaluation of* $\mathfrak{G}$ *in a context c is written* $val \models \mathfrak{G}[c]$*, and it is inductively defined as follows:* $val \models \mathfrak{G}[c] :\Longleftrightarrow$

- $val(\kappa(v)) = \mathtt{tt}$ *for each* $v \in V \cap area(c)$ *(vertex condition), and*
- $val \not\models \mathfrak{G}[c']$ *for each* $c' \in Cut \cap area(c)$ *(cut condition: iteration over* $Cut \cup \{\top\}$*)*

For $val \models \mathfrak{G}[\top]$ *we write* $val \models \mathfrak{G}$ *and say that* $\mathfrak{G}$ *is* valid for *val resp. val* is a model *for* $\mathfrak{G}$*. If we have graphs* $\mathfrak{G}_1$*,* $\mathfrak{G}_2$ *such that* $val \models \mathfrak{G}_2$ *for each valuation val, we write* $\mathfrak{G}_1 \models \mathfrak{G}_2$.

Next, the calculus for formal Alpha graphs will be provided. The rules have already been informally presented in the introduction. We have now the possibility to describe the rules in a mathematically precise manner. Here are the appropriate mathematical definitions:

- **Erasure and Insertion**
  We first provide a general definition for inserting and erasing a subgraph.
  Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph which contains the subgraph $\mathfrak{G}_0 := (V_0, \top_0, Cut_0, area_0, \kappa_0)$. Let $\mathfrak{G}' := (V', \top', Cut', area', \kappa')$ be defined as follows:
  - $V' := V \backslash V_0$, $\top' := \top$ and $Cut' := Cut \backslash Cut_0$,
  - $area'(d) := \begin{cases} area(d) & d \neq \top_0 \\ area(d) \backslash (V_0 \cup Cut_0) & d = \top_0 \end{cases}$.
  - $\kappa' := \kappa|_{V'}$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the subgraph* $\mathfrak{G}_0$ *from the context* $\top_0$, and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting the graph* $\mathfrak{G}_0$ *into the context* $\top_0$. The rules 'erasure' and 'insertion' are restrictions of the definition above:
  Let $\mathfrak{G}$ be a graph and let $\mathfrak{G}_0$ be a subgraph of $\mathfrak{G}$ with $c := ctx(\mathfrak{G}_0)$, and let $\mathfrak{G}'$ be obtained from $\mathfrak{G}$ by erasing $\mathfrak{G}_0$ from the context c. If c is positive, then $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing* $\mathfrak{G}_0$ *from a positive context,* and if c is negative, than $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *inserting* $\mathfrak{G}_0$ *into a negative context.*
- **Iteration and Deiteration**
  Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph which contains the subgraph $\mathfrak{G}_0 := (V_0, \top_0, Cut_0, area_0, \kappa_0)$, and let c be a context with $c \notin Cut_0$.
  Let $\mathfrak{G}' := (V', \top', Cut', area', \kappa')$ be the following graph:

- $V' := V \times \{1\} \cup V_0 \times \{2\}$, $\top' := \top$ and $Cut' := Cut \times \{1\} \cup Cut_0 \times \{2\}$.
- $area'$ is defined as follows:
  for $(d,i) \in Cut'$ and $d \neq c$ let $area'((d,i)) := area(d) \times \{i\}$, and let $area'((c,1)) := area(c) \times \{1\} \cup area_0(\top_0) \times \{2\}$.
- $\kappa'((k,i)) := \kappa(k)$ for all $(k,i) \in V'$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *iterating the subgraph* $\mathfrak{G}_0$ *into the context* c and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *deiterating the subgraph* $\mathfrak{G}_0$ *from the context* c.

– **Double Cuts**

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph and $c_1, c_2 \in Cut$ with $area(c_1) = \{c_2\}$. Let $c_0 := ctx(c_1)$ (i.e., $c_1 \in area(c_0)$) and set $\mathfrak{G}' := (V, \top, Cut', area', \kappa)$ with

- $Cut' := Cut \backslash \{c_1, c_2\}$
- $area'(d) := \begin{cases} area(d) \text{ for } d \neq c_0 \\ area(c_0) \cup area(c_2) \text{ for } d = c_0 \end{cases}.$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the double cuts* c_1, c_2 and $\mathfrak{G}$ is derived from $\mathfrak{G}'$ by *inserting the double cuts* c_1, c_2.

Based on the calculus, we can now define the syntactical entailment relation.

Definition 15. *Let* $\mathfrak{G}_a$, $\mathfrak{G}_b$ *be two graphs. Then* $\mathfrak{G}_b$ can be derived from $\mathfrak{G}_a$ *(which is written* $\mathfrak{G}_a \vdash \mathfrak{G}_b$*), if there is a finite sequence* $(\mathfrak{G}_1, \mathfrak{G}_2, \ldots, \mathfrak{G}_n)$ *with* $\mathfrak{G}_a = \mathfrak{G}_1$ *and* $\mathfrak{G}_b = \mathfrak{G}_n$ *such that each* $\mathfrak{G}_{i+1}$ *is derived from* $\mathfrak{G}_i$ *by applying one of the rules of the calculus. The sequence is called* a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$. *Two graphs* $\mathfrak{G}_1, \mathfrak{G}_2$ *with* $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ *and* $\mathfrak{G}_2 \vdash \mathfrak{G}_1$ *are said to be* provably equivalent.

If $\mathfrak{H} := \{\mathfrak{G}_i \mid i \in I\}$ *is a (possibly empty) set of graphs, then* a graph $\mathfrak{G}$ can be derived from $\mathfrak{H}$ *if there is a finite subset* $\{\mathfrak{G}_1, \ldots, \mathfrak{G}_n\} \subseteq \mathfrak{H}$ *with* $\mathfrak{G}_1 \ldots \mathfrak{G}_n \vdash \mathfrak{G}$ *(remember that* $\mathfrak{G}_1 \ldots \mathfrak{G}_n$ *is the juxtaposition of* $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$*).*

Before we start with the proof that the calculus is sound and complete, we finally show in this section three simple metalemmata in the sense that they show some *schemata* for proofs with EGs, i.e., they are derived 'macro'-rules.

All rules in the calculus which are applied in a context only depend on whether the context is positive or negative. In particular if a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$ is given, this proof can be carried out in arbitrary positive contexts. This yields immediately the following lemma, which can be found in [So97].

Lemma 3 (Cut-And-Paste-Theorem).

Let $\mathfrak{G}_a \vdash \mathfrak{G}_b$ *for two graphs* $\mathfrak{G}_a$, $\mathfrak{G}_b$. *It follows:*

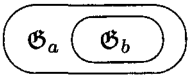
– *If* $\mathfrak{G}_a$ *is a subgraph of* $\mathfrak{G}$ *in a pos. context, then* $\mathfrak{G}_a$ *may be replaced by* $\mathfrak{G}_b$.
– *If* $\mathfrak{G}_b$ *is a subgraph of* $\mathfrak{G}$ *in a neg. context, then* $\mathfrak{G}_b$ *may be replaced by* $\mathfrak{G}_a$.

In particular we have that derivable graphs $\mathfrak{G}_0$ (i.e., graphs with $\vdash \mathfrak{G}_0$) can be inserted into arbitrary contexts of arbitrary graphs.

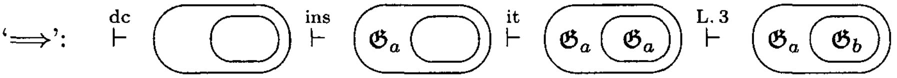
From this lemma we obtain the graph version of the well known deduction theorem.

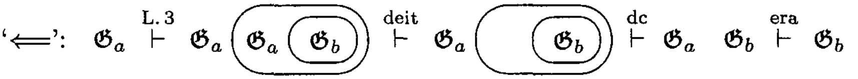
Lemma 4 (Deduction Theorem).

Let $\mathfrak{G}_a$, $\mathfrak{G}_b$ be graphs. Then $\mathfrak{G}_a \vdash \mathfrak{G}_b \iff \vdash$



Proof: We show both directions separately.





□

The following lemma is quite obvious:

Lemma 5. *Let $\mathfrak{G}$, $\mathfrak{G}_a$, $\mathfrak{G}_b$ be graphs with $\mathfrak{G} \vdash \mathfrak{G}_a$ and $\mathfrak{G} \vdash \mathfrak{G}_b$. Then $\mathfrak{G} \vdash \mathfrak{G}_a\ \mathfrak{G}_b$.*

Proof: $\mathfrak{G} \overset{\text{it}}{\vdash} \mathfrak{G}\ \mathfrak{G} \overset{\text{L. 3}}{\vdash} \mathfrak{G}_a\ \mathfrak{G} \overset{\text{L. 3}}{\vdash} \mathfrak{G}_a\ \mathfrak{G}_b$ □

9 Soundness and Completeness

In this chapter we will show that the rules we presented in Sec. 8 are sound and complete with respect to the given semantics. In the first section, we will prove the soundness of the calculus, in the next section we will prove its completeness.

9.1 Soundness

Most of the rules modify only the area of one specific context c (for example, the rule 'erasure' removes a subgraph from the area of a positive context). If a graph $\mathfrak{G}'$ is derived from a graph $\mathfrak{G}$ by applying one of these rules (i.e., by modifying a context c in the graph $\mathfrak{G}$), $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for c. As it has to be shown that no rule can transform a valid graph into a nonvalid one, the following theorem is the basis for proving the soundness of most rules.

Theorem 1 (Main Lemma for Soundness).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ and $\mathfrak{G}' := (V', \top', Cut', area', \kappa')$ be graphs and let $f = f_V \dot{\cup} f_{Cut}$ be an isomorphism from $\mathfrak{G}$ to $\mathfrak{G}'$ except for $c \in Cut \cup \{\top\}$ and $c' \in Cut' \cup \{\top'\}$. Let $val : \mathcal{P} \mapsto \{\mathtt{ff}, \mathtt{tt}\}$ be a valuation. Let $P(d)$ be thefollowing property for Cuts $d \in Cut \cup \{\top\}$:

- *If d is positive and $val \models \mathfrak{G}[d]$, then $val \models \mathfrak{G}'[f(d)]$, and*
- *If d is negative and $val \not\models \mathfrak{G}[d]$, then $val \not\models \mathfrak{G}'[f(d)]$.*

If P holds for c, then P holds for each $d \in Cut \cup \{\top\}$ with $d \not< c$. In particular $v \models \mathfrak{G}'$ follows from $v \models \mathfrak{G}$.

Proof: We set $D := \{d \in Cut \cup \{\top\} \mid d \not< c\}$. D is a tree such that for each $d \in D$ with $d \neq c$ and each $e \in Cut \cup \{\top\}$ with $e < d$ we have $e \in D$. For this reason we can carry out the proof by induction over D. As c satisfies P, it is sufficient to carry out the induction step for $d \neq c$. So let $d \in D$, $d \neq c$ be a context such that $P(e)$ holds for all cuts $e \in area(d) \cap Cut$.

First Case: d is positive and $val \models \mathfrak{G}[d]$.

We have to check the vertex- and cut-conditions for $f(d)$. We start with the vertex conditions for $f(d)$, i.e., for vertices $v' \in V'$ with $ctx'(v') = f(d)$.

For each $v \in V$ with $ctx(v) = d$, it holds $\kappa(v) = \kappa'(f(v))$, hence

$$val(\kappa(v)) = \mathtt{tt} \Longleftrightarrow val(\kappa'(f(v))) = \mathtt{tt}.$$

As f_V is a bijection from $area(d) \cap V$ to $area'(f(d)) \cap V'$, we gain the following: All vertex conditions in d hold iff all vertex conditions in $f(d)$ hold.

As we have $val \models \mathfrak{G}[d]$, we get that $val \not\models \mathfrak{G}[e]$ for all cuts $e \in area(d)$. These cuts are negative and are mapped bijectively to the cuts $e' \in area(f(d))$. As they are negative, we conclude from the induction hypothesis or the presupposition (for $e = c$) that $val \not\models \mathfrak{G}'[f(e)]$ for all cuts $e \in area(d)$, i.e., $val \not\models \mathfrak{G}'[e']$ for all cuts $e' \in area'(f(d))$.

As we have checked all vertex- and cut-conditions for $f(d)$, we get $val \models \mathfrak{G}'[f(d)]$.

Second Case: d is negative and $val \not\models \mathfrak{G}[d]$.

This is shown analogously to the first case. □

With this lemma, we can prove the correctness of the rules. Due to space limitations, the prove will only be carried out for the rules 'iteration' and 'deiteration'. The other rules can be handled similarly.

Lemma 6 (Iteration and Deiteration are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are graphs, val is a valuation with $val \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'iteration' or 'deiteration', then $val \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G}_0 := (V_0, \top_0, Cut_0, area_0, \kappa_0)$ be the subgraph of $\mathfrak{G}$ which is iterated into the context $c \leq ctx(\mathfrak{G}_0)$, $c \notin Cut_0$. We use the mathematical notation which was given in Sec. 8. In particular, $(c, 1)$ is the context in $\mathfrak{G}'$ which corresponds to the context c in $\mathfrak{G}$. There are two cases to consider:

First Case: $val \models \mathfrak{G}_0$. From this we conclude $val \models \mathfrak{G}[c] \iff val \models \mathfrak{G}'[(c,1)]$. As $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for $c \in Cut \cup \{\top\}$ and $(c, 1) \in Cut' \cup \{\top'\}$, Lem. 1 can be applied now. This yields

$$val \models \mathfrak{G} \iff val \models \mathfrak{G}' \ . \qquad (*)$$

Second Case: $val \not\models \mathfrak{G}_0$. This yields $val \not\models \mathfrak{G}[\top_0]$ and $val \not\models \mathfrak{G}'[(\top_0, 1)]$. As $\mathfrak{G}$ and $\mathfrak{G}'$ are isomorphic except for $\top_0 \in Cut \cup \{\top\}$ and $(\top_0, 1) \in Cut' \cup \{\top'\}$, Lemma 1 can be applied now. This yields again $(*)$.

The direction '$\Longrightarrow$' of $(*)$ yields the correctness of the iteration-rule. The opposite direction '$\Longleftarrow$' of $(*)$ yields the correctness of the deiteration-rule. □

Lemma 7 (Erasure and Insertion are Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are graphs, v is a valuation with $val \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying one of the rules 'erasure' or 'insertion', then $val \models \mathfrak{G}'$.

Lemma 8 (Double Cut is Sound).

If $\mathfrak{G}$ and $\mathfrak{G}'$ are graphs, val is a valuation with $val \models \mathfrak{G}$ and $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by applying the rule 'double cut', then $val \models \mathfrak{G}'$.

From the preceding lemmata the soundness of the calculus follows immediately:

Theorem 2 (Soundness of the Alpha-Calculus).

Two formal Alpha graphs $\mathfrak{G}$, $\mathfrak{G}'$ satisfy $\mathfrak{G} \vdash \mathfrak{G}' \Longrightarrow \mathfrak{G} \models \mathfrak{G}'$.

9.2 Completeness

As the empty sheet of assertion is always true, the graph $\boxed{}$ is always false. This leads to the following definition and lemmata.

Definition 16. *A set $\mathfrak{H}$ of graphs is called* consistent *if $\mathfrak{H} \vdash \boxed{}$ does not hold. A graph $\mathfrak{G}$ is called* consistent *if $\{\mathfrak{G}\}$ is consistent.*

Lemma 9. *A set $\mathfrak{H}$ of graphs is not consistent if and only if $\mathfrak{H} \vdash \mathfrak{G}'$ for each graph $\mathfrak{G}'$.*

Proof: Only the direction '$\Longrightarrow$' has to be proven. So let $\mathfrak{G}_1, \ldots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \ldots \mathfrak{G}_n \vdash \boxed{}$. Let $\mathfrak{G}$ be the juxtaposition of $\mathfrak{G}_1, \ldots, \mathfrak{G}_n$. We conclude:

$$\mathfrak{G} \vdash \boxed{} \overset{\text{T. 4}}{\iff} \vdash \boxed{\mathfrak{G}\ \boxed{\boxed{}}} \overset{\text{dc}}{\iff} \vdash \boxed{\mathfrak{G}} \overset{\text{ins}}{\Longrightarrow} \vdash \boxed{\mathfrak{G}\ \boxed{\mathfrak{G}'}} \overset{\text{T. 4}}{\iff} \mathfrak{G} \vdash \mathfrak{G}' \overset{\text{Def.} \vdash}{\iff} \mathfrak{H} \vdash \mathfrak{G}'$$

□

Lemma 10. *Let $\mathfrak{G}$ be a graph and let $\mathfrak{H}$ be a set of graphs. Then we have*

$$\mathfrak{H} \vdash \mathfrak{G} \iff \mathfrak{H} \cup \{\boxed{\mathfrak{G}}\} \vdash \boxed{} \quad \textit{and} \quad \mathfrak{H} \vdash \boxed{\mathfrak{G}} \iff \mathfrak{H} \cup \{\mathfrak{G}\} \vdash \boxed{} \ .$$

Proof of the first equivalence (the second is shown analogously):

$\mathfrak{H} \vdash \mathfrak{G} \overset{\text{Def.} \vdash}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \mathfrak{G}$

$\overset{\text{T. 4}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\vdash \boxed{\mathfrak{G}_1 \dots \mathfrak{G}_n \ \boxed{\mathfrak{G}}}$

$\overset{\text{dc}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\vdash \boxed{\mathfrak{G}_1 \dots \mathfrak{G}_n \ \boxed{\mathfrak{G}} \ \boxed{\boxed{}}}$

$\overset{\text{T. 4}}{\iff}$ there are $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \boxed{\mathfrak{G}} \vdash \boxed{}$

$\overset{\text{Def.} \vdash}{\iff} \mathfrak{H} \cup \{\boxed{\mathfrak{G}}\} \vdash \boxed{}$

□

Consistent sets of graphs can be extended to maximal (with respect to $\subseteq$) consistent sets graphs, which have canonically given valuations satisfying them. This will be elaborated with the next lemmata.

Lemma 11. *Let $\mathfrak{H}$ be a maximal consistent set of graphs. Then:*

1. *Either $\mathfrak{H} \vdash \mathfrak{G}$ or $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$ for each graph $\mathfrak{G}$.*
2. *$\mathfrak{H} \vdash \mathfrak{G} \iff \mathfrak{G} \in \mathfrak{H}$ for each graph $\mathfrak{G}$.*
3. *$\mathfrak{G}_1\ \mathfrak{G}_2 \in \mathfrak{H} \iff \mathfrak{G}_1 \in \mathfrak{H}$ and $\mathfrak{G}_2 \in \mathfrak{H}$ for all graphs $\mathfrak{G}_1, \mathfrak{G}_2$.*

Proof: It is easy to see that $\mathfrak{G} \boxed{\mathfrak{G}} \vdash \boxed{}$. Hence if $\mathfrak{H}$ is consistent, $\mathfrak{H} \vdash \mathfrak{G}$ and $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$ cannot hold both. Now we can prove all propositions of this lemma:

1. Assume $\mathfrak{H} \nvdash \mathfrak{G}$ for a graph $\mathfrak{G}$. Lem. 10 yields that $\mathfrak{H} \cup \{\boxed{\mathfrak{G}}\}$ is consistent. As $\mathfrak{H}$ is maximal, we conclude that $\boxed{\mathfrak{G}} \in \mathfrak{H}$, hence we have $\mathfrak{H} \vdash \boxed{\mathfrak{G}}$.
2. Let $\mathfrak{H} \vdash \mathfrak{G}$. As $\mathfrak{H}$ is consistent, we get that $\mathfrak{H} \nvdash \boxed{\mathfrak{G}}$. So Lem. 10 yields that $\mathfrak{H} \cup \{\mathfrak{G}\}$ is consistent. As $\mathfrak{H}$ is maximal, we conclude $\mathfrak{G} \in \mathfrak{H}$.
3. Follows immediately from 1. and 2. □

Lemma 12. *Let $\mathfrak{H}$ be a consistent set of graphs. Then there is a maximal set $\mathfrak{H}'$ of graphs with $\mathfrak{H}' \supseteq \mathfrak{H}$.*

Proof: Let $\mathfrak{G}_1, \mathfrak{G}_2, \mathfrak{G}_3, \ldots$ be an enumeration of all graphs.We define inductively a set of graphs $\mathfrak{H}_i$ for each $i \in \mathbb{N}$. We start with setting $\mathfrak{H}_1 := \mathfrak{H}$. Assume now that $\mathfrak{H}_i \supseteq \mathfrak{H}$ is defined and consistent.

If $\mathfrak{H}_i \vdash \boxed{\mathfrak{G}_i}$ does not hold, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\mathfrak{G}_i\}$ is consistent due to Lem. 10.

Otherwise, if $\mathfrak{H}_i \vdash \boxed{\mathfrak{G}_i}$ holds, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\boxed{\mathfrak{G}_i}\}$ is consistent.

Now $\mathfrak{H}' := \bigcup_{n \in \mathbb{N}} \mathfrak{H}'_n$ is obviously a consistent maximal graph set with $\mathfrak{H}' \supseteq \mathfrak{H}$. □

Theorem 3. *Let $\mathfrak{H}$ be a maximal consistent set of graphs. Then there exists a canonically given valuation val such that $val \models \mathfrak{G}$ for each graph $\mathfrak{G} \in \mathfrak{H}$.*

Proof: Let us first define a graph which states the propositional variable P_i. We set $\mathfrak{G}(P_i) := (\{v\}, \top, \emptyset, \emptyset, \{(v, P_i)\})$ with an arbitrary vertex v. Let $val : \mathcal{P} \mapsto \{\mathtt{ff}, \mathtt{tt}\}$ be defined as follows: $val(P_i) := \mathtt{tt} : \iff \mathfrak{H} \vdash \mathfrak{G}(P_i)$.

Now let $\mathfrak{G}' := (V, \top, Cut, area, \kappa)$ be a formal Alpha graph. We show

$$val \models \mathfrak{G}'[c] \iff \mathfrak{H} \vdash \mathfrak{G}'[c] \tag{10}$$

for each $c \in Cut \cup \{\top\}$. The proof is done by induction over $Cut \cup \{\top\}$. So let $c \in Cut \cup \{\top\}$ be a cut such that (10) holds for each $d < c$. We have:

$$\begin{aligned}
val \models \mathfrak{G}'[c] \quad &\overset{\text{Def. evaluation}}{\iff} \quad val(\kappa(v)) = \mathtt{tt} \text{ for each } v \in V \cap area(c) \\
&\qquad\qquad \text{and } val \not\models \mathfrak{G}[d] \text{ for each } d \in Cut \cap area(d) \\
&\overset{\text{Def. val and Ind.Hyp.}}{\iff} \quad \mathfrak{H} \vdash \mathfrak{G}(\kappa(v)) \text{ for each } v \in V \cap area(c) \\
&\qquad\qquad \text{and } \mathfrak{H} \not\vdash \mathfrak{G}'[d] \text{ for each } d \in Cut \cap area(d) \\
&\overset{\text{L. 11}}{\iff} \quad \mathfrak{G}(\kappa(v)) \in \mathfrak{H} \text{ for each } v \in V \cap area(c) \\
&\qquad\qquad \text{and } \boxed{\mathfrak{G}'[d]} \in \mathfrak{H} \text{ for each } d \in Cut \cap area(d) \\
&\overset{\text{L. 11}}{\iff} \quad \mathfrak{G}[c] \in \mathfrak{H}
\end{aligned}$$

As we have $\mathfrak{G} = \mathfrak{G}[\top]$, applying (10) to $c := \top$ yields $val \models \mathfrak{G}' \iff \mathfrak{H} \vdash \mathfrak{G}'$. □

Now we are prepared to prove the completeness of the calculus.

Theorem 4 (Completeness of the Calculus).

Two formal Alpha graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ satisfy $\mathfrak{G}_1 \models \mathfrak{G}_2 \implies \mathfrak{G}_1 \vdash \mathfrak{G}_2$.

Proof: Assume that $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ does not hold. Then Cor. 10 yields that $\mathfrak{G}_1 \ \boxed{\mathfrak{G}_2}$ is consistent. According to the last lemma, let be $\mathfrak{H}$ be a maximal consistent set of graphs which includes this graph. Due to Thm. 3, $\mathfrak{H}$ has a canonically given valuation *val*. This valuation is in particular a model for $\mathfrak{G}_1 \ \boxed{\mathfrak{G}_2}$. So v is a model for $\mathfrak{G}_1$, but not for $\mathfrak{G}_2$, which is a contradiction to the assumption. □

References

[Ba93] John Barwise: Heterogenous reasoning, in G. W. Mineau, B. Moulin, J. F. Sowa (Eds.): Conceptual Graphs for Knowledge Representation. LNAI 699, Springer Verlag, Berlin–New York 2000, 64–74.

[Bu91] R. W. Burch: A Peircean Reduction Theses: The Foundations of Topological Logic. Texas Tech University Press, 1991.

[Da00] F. Dau: Negations in Simple Concept Graphs, in: B. Ganter, G. W. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.

[Da01] F. Dau: Concept Graphs and Predicate Logic, in: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base. LNAI 2120, Springer Verlag, Berlin–New York 2001, 72–86.

[Da02] F. Dau: *An Embedding of Existential Graphs into Concept Graphs with Negations.* In: D. Corbett, U: Priss (Eds.): Conceptual Structures: Integration and Interfaces, LNAI, Springer Verlag, Berlin–Heidelberg 2002.

[Da03] F. Dau: *The Logic System of Concept Graphs with Negation (And Its Relationship to Predicate Logic).* LNAI, Vol. 2892, Springer, Berlin–Heidelberg–New York, 2003.

[Ha95] E. M. Hammer, Logic and Visual Information. CSLI Publications, Stanford, California, 1995.

[Ha98] E. M. Hammer, Semantics for Existential Graphs. Journal Philosohpical Logic, Vol. 27, 1998, 489–503.

[HS02] J. Howse, F. Molina, S. Shin, J. Taylor: On Diagram Tokens and Types

[Pe98] C. S. Peirce: Reasoning and the Logic of Things. The Cambridge Conferences Lectures of 1898. Ed. by K. L. Kremer, Harvard Univ. Press, Cambridge 1992.

[PS09] C. S. Peirce, J. F. Sowa: Existential Graphs: MS 514 by Charles Sanders Peirce with commentary by John F. Sowa
`http://users.bestweb.net/~sowa/peirce/ms514.htm`

[Pe03] C. S. Peirce: Existential graphs. 1903. Partly published in the collected papers of Peirce. Complete german translation in:
Helmut Pape: Charles Sanders Peirce: Phänomen und Logik der Zeichen. Suhrkamp Taschenbuch Wissenschaft, 1983.

[Ro73] D. D. Roberts: The Existential Graphs of Charles S. Peirce. Mouton, The Hague, Paris, 1973.

[Ro92] D. D. Roberts: The Existential Graphs. Computers Math. Applic., Vol. 23, No. 6–9, 1992, 639–63.

[Sh94] S. Shin: The Logical Status of Diagrams. Cambridge University Press, 1994.

[Sh99] S. Shin: Reconstituting Beta Graphs into an Efficacious System. Journal of Logic, Language and Information, Vol. 8, No. 3, July 1999.

[Sh01] S. Shin: The Iconic Logic of Peirce's Graphs. Bradford Book, Massachusetts, 2002.

[So84] J. F. Sowa: Conceptual Structures: Information Processing in Mind and Machine. The System Programming Series. Adison-Wesley, Reading 1984.

[So92] J. F. Sowa: Conceptual Graphs Summary, in: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): Conceptual Structures: current research and practice, Ellis Horwood, 1992, 3–51.

[So97] J. F. Sowa: Logic: Graphical and Algebraic, Manuskript, Croton-on-Hudson 1997.

Concept-Based Data Mining with Scaled Labeled Graphs

Bernhard Ganter, Peter A. Grigoriev, Sergei O. Kuznetsov, and
Mikhail V. Samokhin

Technische Universität Dresden
All-Russian Institute for Scientific and Technical Information

Abstract. Graphs with labeled vertices and edges play an important role in various applications, including chemistry. A model of learning from positive and negative examples, naturally described in terms of Formal Concept Analysis (FCA), is used here to generate hypotheses about biological activity of chemical compounds. A standard FCA technique is used to reduce labeled graphs to object-attribute representation. The major challenge is the construction of the context, which can involve ten thousands attributes. The method is tested against a standard dataset from an ongoing international competition called Predictive Toxicology Challenge (PTC).

1 Introduction

In [1] we introduced a general construction based on a semilattice of object description, which we called *pattern structure.* An example that we used was related to a lattice on sets of labeled graphs. In general, pattern structures are naturally reduced to formal contexts. In this paper we present a practical data mining approach which uses JSM or concept-based hypotheses. On the data side we use a standard FCA technique, called ordinal scaling [2] for the reduction of labeled graphs to formal contexts. We consider a chemical application in Predictive Toxicology and compare the results to those obtained with the same learning model, but different representation language which used predefined descriptors (attributes) for describing chemical compounds.

2 A Learning Model

2.1 Pattern Structures

In [1] we showed how such an approach is linked to the general FCA framework [2]. In [3] and in [4] we showed how this approach is related to standard machine learning models such as version spaces and decision trees.

Let G be some set, let $(D, \sqcap)$ be a meet-semilattice and let $\delta : G \to D$ be a mapping. Then $(G, \underline{D}, \delta)$ with $\underline{D} = (D, \sqcap)$ is called a *pattern structure,* provided that the set

$$\delta(G) := \{\delta(g) \mid g \in G\}$$

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 94–108, 2004.

generates a complete subsemilattice $(D_\delta, \sqcap)$ of $(D, \sqcap)$ (e.g., when $(D, \sqcap)$ is complete, or when G is finite), i.e., every subset X of $\delta(G)$ has an infimum $\sqcap X$ in $(D, \sqcap)$ and D_δ is the set of these infima.

If $(G, \underline{D}, \delta)$ is a pattern structure, the derivation operators are defined as

$$A^\diamond := \sqcap_{g \in A} \delta(g) \qquad \text{for } A \subseteq G$$

and

$$d^\diamond := \{g \in G \mid d \sqsubseteq \delta(g)\} \qquad \text{for } d \in D.$$

The elements of D are called *patterns.* The natural order on them is given, as usual, by

$$c \sqsubseteq d :\Longleftrightarrow c \sqcap d = c,$$

and is called the *subsumption* order.

The operators $(.)^\diamond$ obviously make a Galois connection between the power set of G and $(D, \sqsubseteq)$. The pairs (A, d) satisfying

$$A \subseteq G, \quad d \in D, \quad A^\diamond = d, \quad \text{and} \quad A = d^\diamond$$

are called the *pattern concepts* of $(G, \underline{D}, \delta)$, with extent A and *pattern intent* d. For $a, b \in D$ the *pattern implication* $a \to b$ holds if $a^\diamond \subseteq b^\diamond$. Similarly, for $C, D \subseteq G$ the *object implication* $C \to D$ holds if $C^\diamond \sqsubseteq D^\diamond$.

Since $(D_\delta, \sqcap)$ is complete, there is a (unique) operation $\sqcup$ such that $(D_\delta, \sqcap, \sqcup)$ is a complete lattice. It is given by

$$\sqcup X := \sqcap\{c \in D_\delta \mid \forall_{x \in X}\ x \sqsubseteq c\}.$$

A subset M of D is $\sqcup$-*dense* for $(D_\delta, \sqcap)$ if every element of D_δ is of the form $\sqcup X$ for some $X \subseteq M$. If this is the case, then with

$$\downarrow d := \{e \in D \mid e \sqsubseteq d\}$$

we get

$$c = \sqcup(\downarrow c \cap M) \qquad \text{for every } c \in D_\delta.$$

Of course, $M := D_\delta$ is always an example of a $\sqcup$-dense set.

If M is $\sqcup$-dense in $(D_\delta, \sqcap)$, then the formal context (G, M, I) with I given as $gIm :\Leftrightarrow m \sqsubseteq \delta(g)$ is called a *representation context* for $(G, \underline{D}, \delta)$.

In [1] we proved that for any $A \subseteq G$, $B \subseteq M$ and $d \in D$ the following two conditions are equivalent:

1. (A, d) is a pattern concept of $(G, \underline{D}, \delta)$ and $B = \downarrow d \cap M$.
2. (A, B) is a formal concept of (G, M, I) and $d = \bigsqcup B$.

Thus, the pattern concepts of $(G, \underline{D}, \delta)$ are in 1-1-correspondence with the formal concepts of (G, M, I). Corresponding concepts have the same first components (called *extents*). These extents form a closure system on G and thus a complete lattice, which is isomorphic to the concept lattice of (G, M, I).

2.2 Hypotheses in Pattern Structures

In [5,6,7] we considered a learning model from [8] in terms of Formal Concept Analysis. This model assumes that the cause of a *target property* resides in common attributes of objects that have this property.

For pattern structures this can be formalized as follows. Let $(G, \underline{D}, \delta)$ be a pattern structure together with an external target property ω. As in the case of standard contexts, the set G of all objects is partitioned into three disjoint sets w.r.t. ω: the sets G_+, G_-, G_τ of positive, negative, and undetermined examples, respectively. For positive examples it is known that they have property ω, for negative examples it is known that they do not have property ω, and for undetermined examples it is not known whether they have or do not have ω. This gives three pattern substructures of $(G, \underline{D}, \delta)$: $(G_+, \underline{D}, \delta_+)$, $(G_-, \underline{D}, \delta_-)$, $(G_\tau, \underline{D}, \delta_\tau)$, where δ_ε for $\varepsilon \in \{+, -, \tau\}$ are restrictions of δ to the corresponding sets of examples. For brevity sake, we shall write just δ instead of δ_ε.

A *positive hypothesis* h is defined as a pattern intent of $(G_+, \underline{D}, \delta)$ that is not subsumed by any pattern from $\delta(G_-)$ (for short: not subsumed by any negative example). Formally: $h \in D$ is a positive hypothesis iff

$$h^\diamond \cap G_- = \emptyset \text{ and } \exists A \subseteq G_+ : A^\diamond = h.$$

A *negative hypothesis* is defined accordingly. A hypothesis in the sense of [9,8,7] is obtained as a special case of this definition when $(D, \sqcap) = (2^M, \cap)$ for some set M then we have a standard context (object-attribute) representation.

Hypotheses can be used for classification of undetermined examples as introduced in [8] in the following way. If $g \in G_\tau$ is an undetermined example, then a hypothesis h with $h \sqsubseteq \delta(g)$ is *for the positive classification* of g if h is positive and *for the negative classification* of g if it is a negative hypothesis.

An example $g \in G_\tau$ is classified positively if there is a hypothesis for its positive classification and no hypothesis for the negative classification. g is classified negatively in the opposite case. If there are hypotheses for both positive and negative classification, then some other methods (based on standard statistical techniques) may be applied.

2.3 An Example with Labeled Graphs

Consider a pattern structure based on a given ordered set G of graphs (V, E) with vertex- and edge-labels from the sets $(\mathcal{L}_\mathcal{V}, \preceq)$ and $(\mathcal{L}_\mathcal{E}, \preceq)$. Each labeled graph Γ from G is a quadruple of the form $((V, l), (E, b))$, where V is a set of vertices, E is a set of edges, $l \colon V \to \mathcal{L}_\mathcal{V}$ is a function assigning labels to vertices, and $b \colon E \to \mathcal{L}_\mathcal{E}$ is a function assigning labels to edges. We do not distinguish isomorphic graphs with identical labelings.

The order is defined as follows: For two graphs $\Gamma_1 := ((V_1, l_1), (E_1, b_1))$ and $\Gamma_2 := ((V_2, l_2), (E_2, b_2))$ from G we say that Γ_1 **dominates** Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a **subgraph** of Γ_1) if there exists a one-to-one mapping $\varphi : V_2 \to V_1$ such that it

- respects edges: $(v,w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_2(v) \preceq l_1(\varphi(v))$, $(v,w) \in E_2 \Rightarrow b_2(v,w) \preceq b_1(\varphi(v), \varphi(w))$.

Two small examples are given in Figure 1.

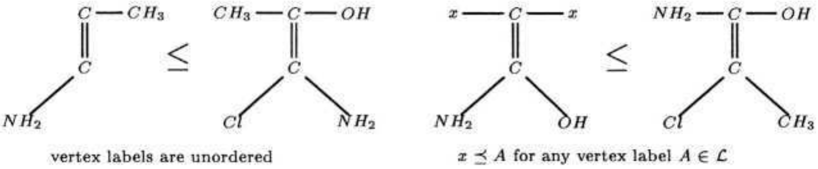


Fig. 1. With $\mathcal{L}_\mathcal{V} = \{\mathrm{C}, \mathrm{NH_2}, \mathrm{CH_3}, \mathrm{OH}, x\}$ we have subgraphs as shown in the diagrams above. In all subsequent examples the label value x will not be admitted and unordered vertex labels will be used

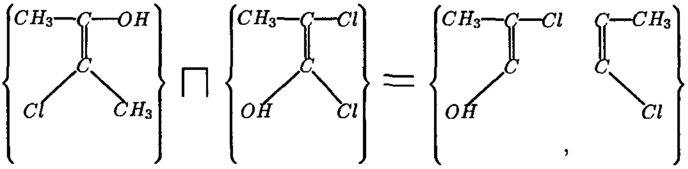
A pattern structure for these graphs then is defined as $(G, \underline{D}, \delta)$, where the semilattice $\underline{D} := (D, \sqcap)$ consists of all sets of subgraphs of graphs from G ("graph sets"), and the meet operation $\sqcap$ on graph sets is defined as follows: For two graphs X and Y from G

$$\{X\} \sqcap \{Y\} := \{Z \mid Zy \leq X, Y, \ \forall Z_* \leq X, Y \ Z_* \not\geq Z\},$$

i.e., $\{X\} \sqcap \{Y\}$ is the set of all maximal common subgraphs of X and Y. The meet of non-singleton sets of graphs is defined as

$$\{X_1, \ldots, X_k\} \sqcap \{Y_1, \ldots, Y_m\} := \mathrm{MAX}_{\leq}(\sqcup_{i,j}(\{X_i\} \sqcap \{Y_j\}))$$

for details see [10,6,1]. Here is an example of applying $\sqcap$ defined above:



To get an example of such a pattern structure, let $G := G_+ \cup G_-$, Where G_+ consists of the first four graphs $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ of Figure 2 and $G_- := \{\Gamma_5, \Gamma_6, \Gamma_7\}$.

The pattern concept lattice of the positive pattern structure $(G_+, \underline{D}, \delta)$ is given in Figure 3.

2.4 Projections and Projected Hypotheses

Since for some pattern structures (e.g., for the pattern structure given by sets of graphs with labeled vertices) even computing subsumption relation may be NP-hard, in [1] we introduced projection operators to approximate pattern structures. A projection (kernel operator) is a mapping $\psi \colon D \to D$ which is

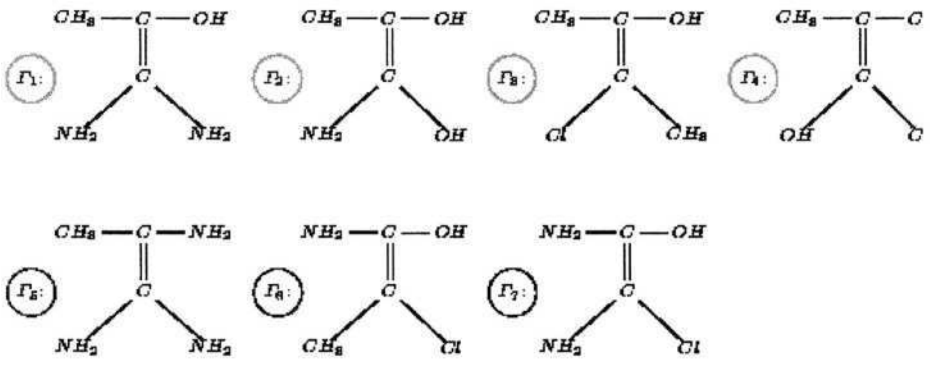


Fig. 2. Seven labeled graphs for a pattern structure.

monotone: if $x \sqsubseteq y$, then $\psi(x) \sqsubseteq \psi(y)$,
contractive: $\psi(x) \sqsubseteq x$, and
idempotent: $\psi(\psi(x)) = \psi(x)$.

Any projection of a complete semilattice $(D, \sqcap)$ is $\sqcap$-preserving, i.e., for any $X, Y \in D$

$$\psi(X \sqcap Y) = \psi(X) \sqcap \psi(Y),$$

which helps us to describe how the lattice of pattern concepts changes when we replace $(G, \underline{D}, \delta)$ by its approximation $(G, \underline{D}, \psi \circ \delta)$. First, we note that $\psi(d) \sqsubseteq \delta(g) \Leftrightarrow \psi(d) \sqsubseteq \psi \circ \delta(g)$. Then, using the basic theorem of FCA (which, in particular allows one to represent every lattice as a concept lattice), we showed how the projected pattern lattice is represented by a context [1]:

Theorem 1 *For pattern structures $(G, \underline{D}, \delta_1)$ and $(G, \underline{D}, \delta_2)$ the following statements are equivalent:*

1. *$\delta_2 = \psi \circ \delta_1$ for some projection ψ of $\underline{D}$.*
2. *There is a representation context (G, M, I) of $(G, \underline{D}, \delta_1)$ and some $N \subseteq M$ such that $(G, N, I \cap (G \times N))$ is a representation context of $(G, \underline{D}, \delta_2)$.*

The properties of projection allow one to relate hypotheses in the original representation with those approximated by a projection. As in [1] we use the term "hypothesis" to those obtained for $(G, \underline{D}, \delta)$ and we refer to those obtained for $(G, \underline{D}, \psi \circ \delta)$ as *ψ-hypotheses*. There is no guarantee that the ψ-image of a hypothesis will be a ψ-hypothesis. In fact, our definition allows that ψ is the "null projection" with $\psi(d) = \mathbf{0}$ for all $d \in D$ (total abandoning of the data with no interesting hypotheses). However, if $\psi(d)$ is a (positive) hypothesis, then $\psi(d)$ is also a (positive) ψ-hypothesis. If we want to look the other way round, we have the following: if $\psi(d)$ is a (positive) ψ-hypothesis, then $\psi(d)^{\diamond\diamond}$ is a (positive) hypothesis [1].

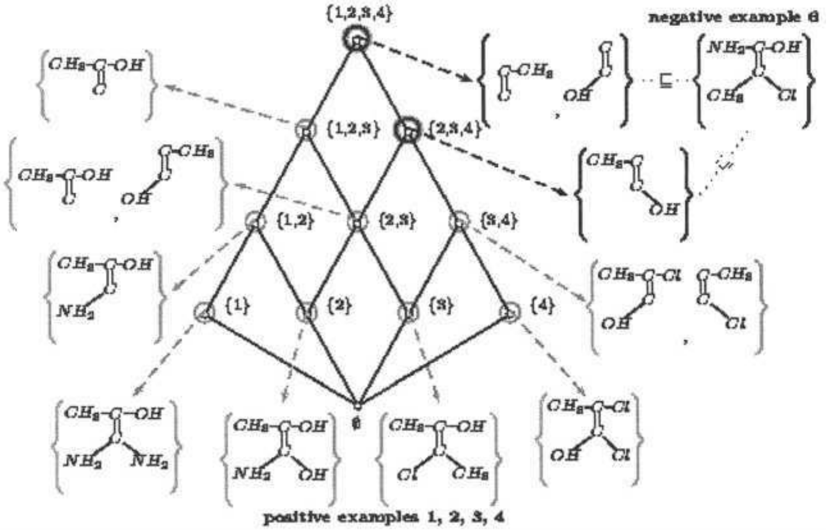


Fig. 3. The pattern concept lattice of the positive pattern structure

The set of all hypothesis-based classifications does not shrink when we pass from d to $\psi(d)$. Formally, if d is a hypothesis for the positive classification of g and $\psi(d)$ is a positive ψ-hypothesis, then $\psi(d)$ is for the positive classification of g.

The above observations show that we can generate hypotheses starting from projections. For example, we can select only those that can be seen in the projected data, which is suggested by the following theorem from [1]:

Theorem 2 *For any projection ψ and any positive hypothesis $d \in D$ the following are equivalent:*

1. *$\psi(d)$ is not subsumed by any negative example.*
2. *There is some positive ψ-hypothesis h such that $h^{\diamond\diamond} \sqsubseteq d$.*

An example is shown in Figure 4). We have used the same data as in Figure 3, but the set D of graph sets was restricted to graphs with less then four vertices.

3 Scaling Labeled Graphs and Their Projections

In this section we shall discuss an application of the learning model, introduced in Section 2 to the problem of bioactivity prediction. In our experiments we will

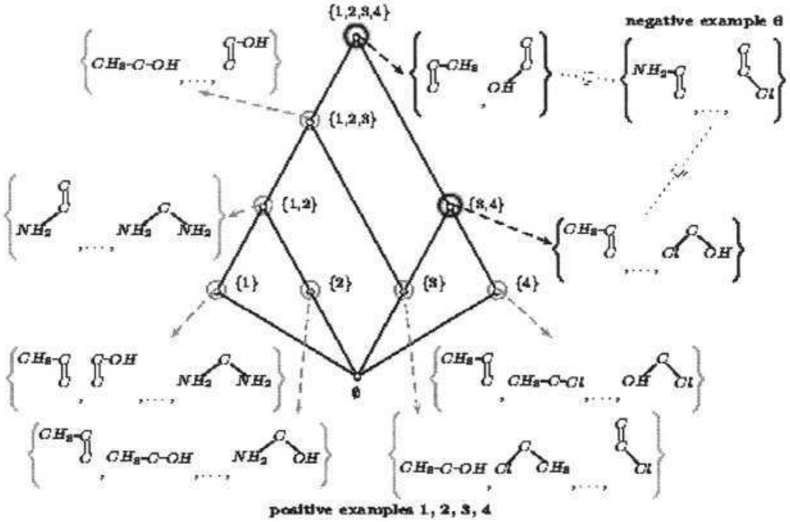


Fig. 4. The lattice of the projected positive pattern structure

use the data from the Predictive Toxicology Challenge (PTC), an ongoing international competition. First, we shall briefly describe the format of PTC (Subsection 3.1). Then we shall discuss k-projections of labeled undirected graphs as a means of approximate representation of graph data.

3.1 Predictive Toxicology Challenge

The program of a workshop on Predictive Toxicology Challenge (PTC) [11], (at the joint 12th European Conference on Machine Learning and the 5th European Conference on Principles of Knowledge Discovery in Databases) consisted in a competition of machine learning programs for generation of hypothetical causes of toxicity from positive and negative examples of toxicity. The organizers (Machine Learning groups of the Freiburg University, Oxford University, and University of Wales) together with toxicology experts (US Environmental Protection Agency, US National Institute of Environmental and Health Standards) provided participants with training and test datasets.

The training dataset consisted of descriptions of 185 molecular graphs of 409 chemical compounds with indication of whether a compound is toxic or not for a particular sex/species group out of four possible groups: male mice, female mice, male rats and female rats. For each group there were about 120 to 150 positive examples and 190 to 230 negative examples of toxicity. The test dataset consisted of 185 substances for which forecasts of toxicity should be made. Twelve research groups (world-wide) participated in PTC, each with up to 4 prediction models for every sex/species group.

3.2 Representation Contexts for Graph Data

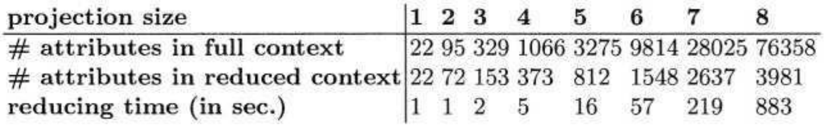
The source data of the PTC datasets are molecular graphs. These are graphs in the sense of Subsection 2.3 above, with labeled Vertices and edges. We cant therefore apply the methods from Section 2. Another view is the following (working directly with a representation context): Think of the source data as a many-valued context with a single many-valued attribute “graph”, assigning to each compound its molecular graph. This many-valued context is (ordinaly) scaled one, such that the attributes of the derived context are all (connected) subgraphs of the graphs under consideration and each graph has its subgraps as attributes.

However, generating this context is a rather time-consuming process. The difficulty of generating all subgraphs is due to costly isomorphism and subgraph isomorphism testing (the latter is an NP-complete problem). There are several well-known algorithms for these problems, e.g., that of B. D. McKay [12] for isomorphism testing and the algorithm of J. R. Ullmann [13] for testing subgraph isomorphism. Since the generation of all subgraphs for an arbitrary labeled graph is a computationally hard task, we use k-projections of initial graphs. The notion of projection introduced in Section 2 for general semilattices can be specified for the lattice on graph sets, e.g., as follows.

Definition 1. *Let $\Gamma = ((V,l),(E,b))$ be a labeled graph. The set $S_\Gamma = \{\Gamma_* = ((V_*,l_*),(E_*,b_*)) \mid \Gamma_*$ is connected, $\Gamma_* \leq \Gamma$, $|V_*| \leq k\}$ is called a k-projection of Γ.*

Thus, k-projection of a labeled graph Γ is a set of all subgraphs (up to isomorphism) of Γ with up to k-size set of vertices. Obviously, k-projection satisfies the properties of the kernel operator. When we use k-projections of graphs, then in the corresponding representation context (G, M, I), the set of objects G is a set of chemical compound names, M is a set of subgraphs of molecular graphs with k or less vertices and gIm means that the graph $m \in M$ is a subgraph of the molecular graph of the compound $g \in G$.

So far, we have generated these contexts for the values of k from 1 up to 8. With the growth of k, the number of attributes in the resulting scaled context becomes very large (thousands of attributes), but reduction of attributes (a standard FCA technique) reduces the size of contexts in several times, see Figure 5.



projection size	1	2	3	4	5	6	7	8
# attributes in full context	22	95	329	1066	3275	9814	28025	76358
# attributes in reduced context	22	72	153	373	812	1548	2637	3981
reducing time (in sec.)	1	1	2	5	16	57	219	883

Fig. 5. PTC dataset: number of attributes in representation contexts before and after attribute reduction

4 QuDA: Qualitative Data Analysis

After the preprocessing step described above, we had 8 datasets. Each of these datasets was an encoding of the original PTC data by means of 1- to 8-projections, respectively. Our goal was to test whether hypotheses obtained for these encodings can be better than those obtained for other encoding schemes proposed by the PTC participants [11]. To do that we have used the QuDA software [14,15].

4.1 A Brief Description of QuDA

History and motivation. *QuDA, a data miners' discovery environment* was developed at the Intellectics group of the Darmstadt University of Technology in 2001-2003. The authors are P. Grigoriev and S. Yevtushenko. This project was started as a "companion" data mining system for the DaMiT tutorial [16] by initiative of the authors.

Usage. QuDA can serve as a personal data mining environment for analyzing mid-sized datasets (up to ten thousands of records). Most of its functionality is also accessible from external applications thus allowing the use of it in integrated solutions. QuDA has an open architecture; it supports scripting and has import/export capabilities for the most commonly used data/model formats.

Functionality. QuDA implements various data mining techniques, including association rule mining, decision tree induction, JSM-reasoning[1], Bayesian learning, and interesting subgroup discovery. It provides also several preprocessing and postprocessing utilities, including data cleaning tools, visualization of attribute distributions, dynamic Hasse diagrams, and a ruleset navigator.

Implementation. QuDA is implemented entirely in Java. The system consists of approximately 1700 classes; the source codes take approximately 3.5Mb.

Availability. QuDA is freeware. It is available for download at the DaMiT tutorial site (http://damit.dfki.de) as well as at its own site at the Intellectics group (http://ki-www2.intellektik.informatik.tu-darmtadt.de/~jsm/QDA).

4.2 Using QuDA for Bioactivity Prediction: Sequential Covering

QuDA implements several variants of the learning model described in Section 2. In our experiments we used the strategy of sequential covering with structural similarities. This procedure does not generate the set of all hypotheses, but generates a subset of it, sufficient to explain every of the examples in the training set.

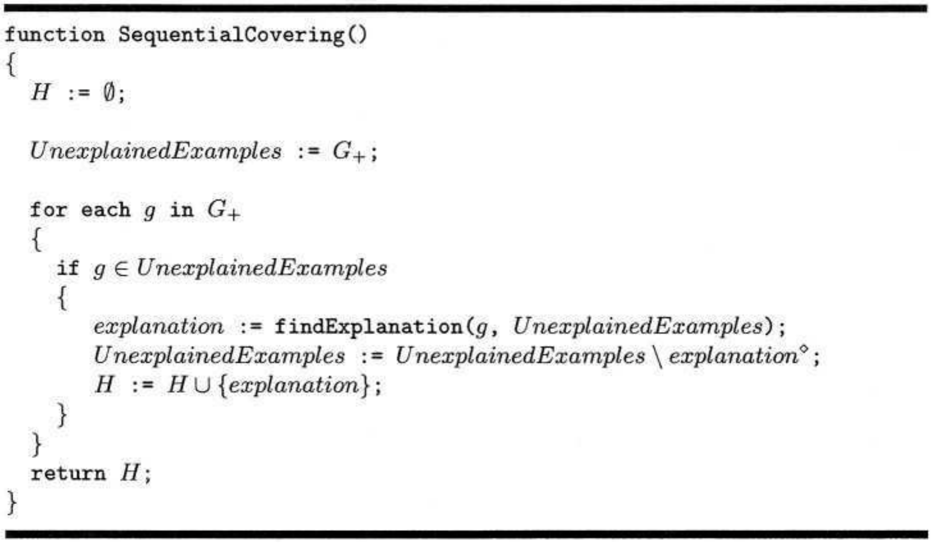
Briefly, this strategy can be described as follows:

1. objects in G_+ are sorted according to some fixed linear order (e.g., in the order they appear in the dataset);

[1] This name stands for a group of lattice-based machine learning methods, including the model described in Section 2.

2. the set of hypotheses H is initialized with an empty set;
3. first object g_+ in G_+, which is not covered by any hypothesis in H is selected;
4. a hypotheses h, covering g_+ is found by generalizing its description with descriptions of other objects in G_+ uncovered so far by *H;*
5. the new-found hypothesis h is added to H and the procedure continues from the step 3 until every object in G_+ is covered by at least one hypothesis in H.

A pseudocode for the main loop of this procedure is given in Figure 6. Figure 7 provides pseudocode for the step 4: finding a hypothesis that explains a particular example, uncovered so far. We provide only the pseudocode for generating *positive* hypotheses. Generating *negative* hypotheses is organized dually.

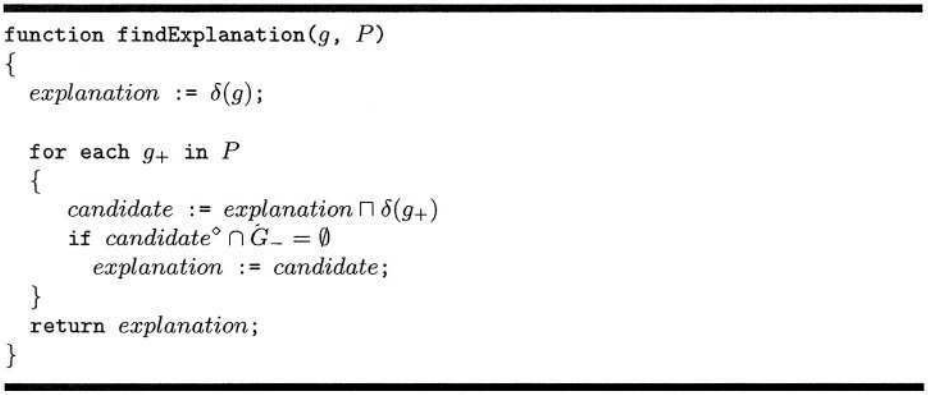


```
function SequentialCovering()
{
    $H$ := $\emptyset$;

    $UnexplainedExamples$ := $G_+$;

    for each $g$ in $G_+$
    {
        if $g \in UnexplainedExamples$
        {
            $explanation$ := findExplanation($g$, $UnexplainedExamples$);
            $UnexplainedExamples$ := $UnexplainedExamples \setminus explanation^{\diamond}$;
            $H$ := $H \cup \{explanation\}$;
        }
    }
    return $H$;
}
```

Fig. 6. Sequential covering: the main loop

Although the sequential covering strategy has a number of obvious drawbacks (most notably – its dependence on the selected order of objects), we decided to use this strategy instead of generating all hypotheses for several reasons:

- it has attractive computational complexity: linear in the number of attributes in the representation context (see Section 2), linear in the number of negative examples (G_-), and quadratic in the number of positive examples (G_+);
- in practical experiments on a number of real-world datasets [15] it has shown classification accuracy and recall comparable to those of the strategy where all hypotheses are generated.



```
function findExplanation($g$, $P$)
{
  $explanation$ := $\delta(g)$;

  for each $g_+$ in $P$
  {
     $candidate$ := $explanation \sqcap \delta(g_+)$
     if $candidate^\diamond \cap \dot{G}_- = \emptyset$
       $explanation$ := $candidate$;
  }
  return $explanation$;
}
```

Fig. 7. Sequential covering: `findExplanation` procedure

We conclude this section with a final remark, which aims at making our results reproducible. The sequential covering strategy naturally depends on the order on objects. This order is used in the main loop to select the next object to find an explanation for; at the same time this order determines the sequence in which objects are used for generalization in the `findExplanation` procedure. In our experiments we have used the order in which objects were presented in the source dataset.

5 Experiments with the PTC Dataset

One of the standard techniques used in machine learning for comparison of the classification algorithms is the so-called ROC-analysis [2]. Here, results obtained by a learning model are represented by a point on (x, y)-plane, where x stays for the relative number of false positive predictions and y stays for the relative number of true positive predictions. The best (usually unattaianble) point is (0,1) and the strait line from (0,0) to (1,1) corresponds to models that are equivalent to random guess under uniform distribution. When the costs of correct classification and misclassification are not known in advance, “best” models correspond to points lying on the convex hull of leftmost points. ROC-diagrams were used in the Predictive Toxicology Challenge to select the best learning models. Here we also use ROC-diagrams to demonstrate the usage of projected pattern structures for bioactivity prediction in comparison to other encoding schemes and/or other learning models.

[2] ROC is the abbreviation for Receiver Operating Characteristic, see [17]

5.1 Projected Pattern Structures “On the ROC”

The results are shown in Figure 8. The following abbreviations are used:

- PR1, PR2, ..., PR8 – the results obtained using 1- to 8-Projection representations, respectively, in combination with sequential covering strategy;
- WAI1, GONZ, KWAI, LEU3 are other “best” models submitted to the Predictive Toxicology Challenge for this animal group.

Note, that the Figure 8 shows both the “old” ROC-curve (composed by LEU3, KWAI, GONZ, and WAI1 models) and the “new” one (composed by LEU3, PR7, PR5, and WAI1 models).

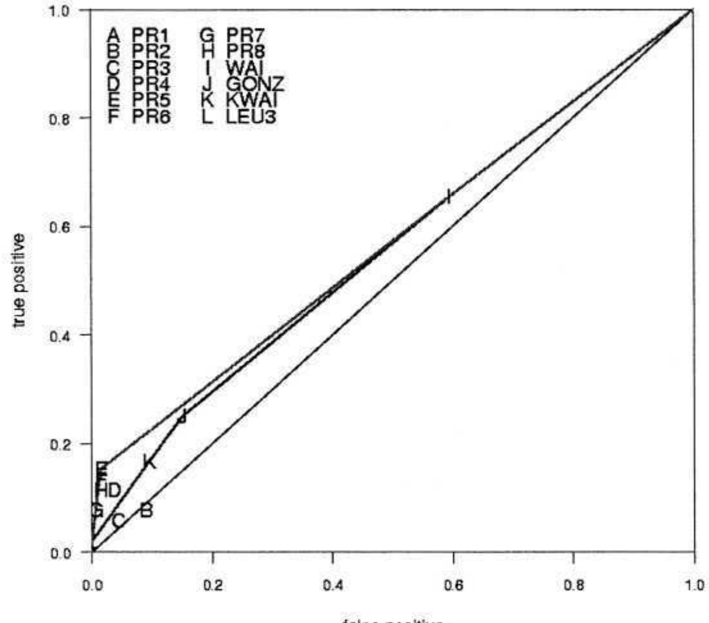


Fig. 8. Projected pattern structures “On the ROC”. Animal group: MR (male rats).

As one can “read” from the ROC-diagram in Figure 8:

- using 1-Projections does not lead to any classifications at all (false positive rate and true positive rate equal zero);
- using 2-Projections results in rather bad classifications: the corresponding point on the ROC-diagram is below the diagonal;

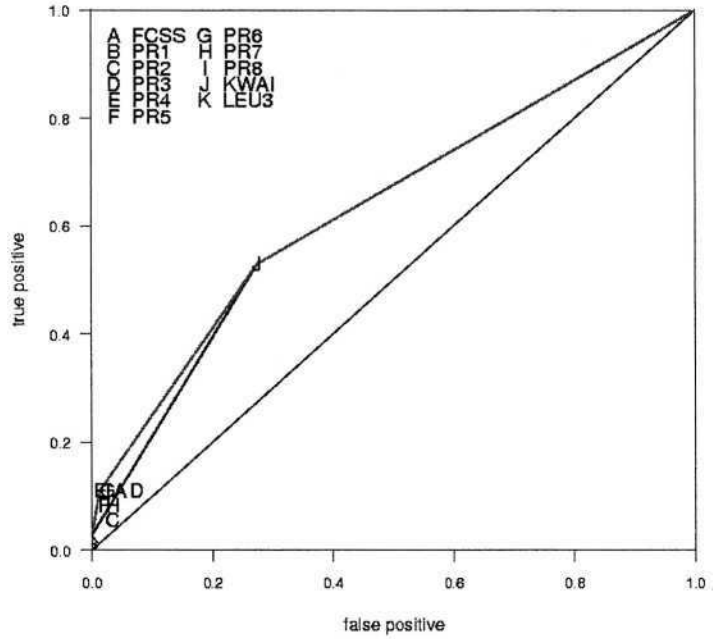


Fig. 9. Projected pattern structures "On the ROC". Animal group: FR (female rats).

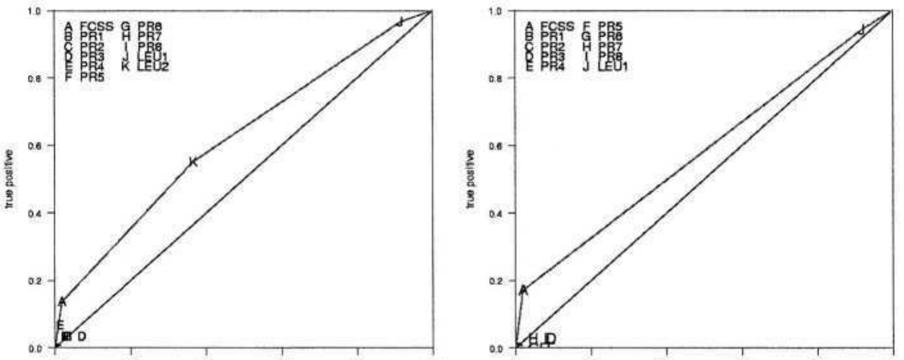


Fig. 10. Projected pattern structures "On the ROC". Animal groups: MM and FM (male and female mice).

- using 3-Projections results in better classifications: the corresponding point on the ROC-diagram is above the diagonal;
- using 4-Projections results in even better classifications: the corresponding point is above the “old” ROC-curve;
- using 5-Projections occurs to be one of the four “new” best strategies: it results in making 8 true positive predictions with only 2 false positive ones;
- using 6-Projections, however, does not result in better classifications: the number of true positives decreases to 6; the number of false positives remains the same;
- using 7-Projections, with 4 true positives and 1 false positives again appears on the “new” ROC-curve;
- using 8-Projections increases the number of true positives to 6, but also increases the number of false positives to 2; this strategy is thus strictly worse than using 5-Projections (assuming positive cost of making a true positive classification);

For the FR group (female rats; see Figure 9) the strategies with 4-, 5-, 6-, and 8-Projections occur above the “old” ROC-curve, without, however, making any of the “old” best models (LEU3 and KWAI) lose their positions.

For the other two animal groups (MM and FM, male and female mice) our strategy did not bring any good results (see Figure 10).

6 Conclusions and Further Work

Our practical result is: *In two animal groups of the four the classification accuracy obtained with molecular graph projections and sequential covering strategy appeared to be among the best known.* In the other two groups, however, this was not the case.

Somewhat more interesting, although expected result is the demonstrated *non-monotonicity of the classification accuracy by k-Projections with the growth of k.* At first glance, this result may seem strange, as increasing the size of projections we increase the amount of information available for the learning algorithm. However, in practice this information growth often results in generating more irrelevant hypotheses and thus, in the decrease of classification accuracy.

The most interesting directions of further research are as follows:

- check the proposed approach on other real-world datasets involving graph representation; these include other datasets from the bioactivity prediction domain as well as datasets from other domains, e.g. web structure mining, ontology-based text mining, etc.;
- incorporate standard machine learning algorithms for feature selection and/or develop specialized ones to overcome classification accuracy decrease when growing the size of projections;
- in our experiments we have used a trivial conflict resolution technique: if a certain object contained both positive and negative hypotheses it remained undefined; other strategies, e.g. voting, may prove more appropriate.

References

1. B. Ganter and S. Kuznetsov. Pattern Structures and Their Projections. In G. Stumme and H. Delugach, editors, *Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01,* volume 2120 of *Lecture Notes in Artificial Intelligence,* pages 129–142. Springer-Verlag, 2001.
2. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical foundations.* Springer-Verlag, Berlin Heidelberg New-York, 1999.
3. B. Ganter and S.O. Kuznetsov. Hypotheses and version spaces. In W. Lex A.de Moor and B.Ganter, editors, *Proc. 10th Int. Conf. on Conceptual Structures, ICCS'01,* volume 2746 of *Lecture Notes in Artificial Intelligence,* pages 83–95. Springer-Verlag, 2003.
4. S.O. Kuznetsov. Machine learning and Formal Concept Analysis. In P. Eklund, editor, *Concept Lattices,* volume 2961 of *Lecture Notes in Artificial Intelligence,* pages 287–312. Springer-Verlag, 2004.
5. S.O. Kuznetsov and V.K. Finn. On a model of learning and classification based on similarity operation. *Obozrenie Prikladnoi i Promyshlennoi Matematiki* **3**, 1:66–90, 1996. in Russian.
6. S.O. Kuznetsov. Learning of Simple Conceptual Graphs from Positive and Negative Examples. In J. Zytkow and J. Rauch, editors, *Proc. Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD'99,* volume 1704 of *Lecture Notes in Artificial Intelligence,* pages 384–392. Springer-Verlag, 1999.
7. B. Ganter and S. Kuznetsov. Formalizing Hypotheses with Concepts. In B. Ganter and G. Mineau, editors, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'00,* volume 1867 of *Lecture Notes in Artificial Intelligence,* pages 342–356. Springer-Verlag, 2000.
8. V.K. Finn. Plausible Reasoning in Systems of JSM Type. *Itogi Nauki i Tekhniki, Seriya Informatika,* 15:54–101, 1991. in Russian.
9. V.K. Finn. On Machine-Oriented Formalization of Plausible Reasoning in the Style of F. Backon–J. S. Mill. *Semiotika i Informatika,* 20:35–101, 1983. in Russian.
10. S.O. Kuznetsov. JSM-method as a machine learning method. *Itogi Nauki i Tekhniki, ser. Informatika,* 15:17–50, 1991. in Russian.
11. C. Helma, R.D. King, S. Kramer, and A. Srinvasan, editors. *Proc. of the Workshop on Predictive Toxicology Challegnge at the 5th Conference on Data Mining and Knowledge Discovery (PKDD'01) Freiburg (Germany),* http://www.predictive-toxicology.org/ptc/, 2001, September 7.
12. B.D. McKay. Practical graph isomorphism. *Congressus Numerantium,* 30:45–87, 1981.
13. J.R. Ullmann. An algorithm for subgraph isomorphism. *J. Assoc. Comput. Mach.,* 23:31–42, 1976.
14. P.A. Grigoriev, S.A. Yevtushenko, and G.Grieser. QuDA, a data miner's discovery enviornment. Technical Report AIDA 03 06, FG Intellektik, FB Informatik, Technische Universität Darmstadt, http://www.intellektik.informatik.tu-darmstadt.de/~peter/QuDA.pdf, September 2003.
15. P. Grigoriev and S. Yevtushenko. JSM-Reasoning as a data mining tool. In *Proceedings of the 8th Russian National Conference on Artificial Intelligence, CAI-2002,* pages 112–122, Moscow, 2002. PhysMathLit. In Russian.
16. DaMiT, the **Da**ta **Mi**ning online **T**utorial. http://damit.dfki.de.
17. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning,* 42:203–231, 2001.

Iceberg Query Lattices for Datalog

Gerd Stumme

Knowledge & Data Engineering Group
Department of Mathematics & Computer Science
University of Kassel, D–34121 Kassel, Germany
stumme@cs.uni-kassel.de,www.kde.cs.uni-kassel.de

Abstract. In this paper we study two orthogonal extensions of the classical data mining problem of mining association rules, and show how they naturally interact. The first is the extension from a propositional representation to datalog, and the second is the condensed representation of frequent itemsets by means of Formal Concept Analysis (FCA). We combine the notion of *frequent datalog queries* with *iceberg concept lattices* (also called *closed itemsets*) of FCA and introduce two kinds of *iceberg query lattices* as condensed representations of frequent datalog queries. We demonstrate that iceberg query lattices provide a natural way to visualize relational association rules in a non-redundant way.

1 Introduction

Mining association rules is a popular knowledge discovery problem. Since the problem was stated [1], various approaches have been proposed for an increased efficiency of rule discovery in very large databases [2,6,8,23,24]. In parallel, researchers have extended the original problem to knowledge representations that are either related to and/or more expressive than the original representation in propositional logic of itemsets. These include for instance generalized association rules [28], or frequent patterns within time series [3].

In this paper, we consider the extension to first order logic as introduced by L. de Haspe and H. Toivonen in [10] and [11]. Instead of considering rules of the form $X \to Y$ where X and Y are sets of attributes (items; e. g., products of a supermarket) which may or may not apply to objects (e. g., to transactions), they consider X and Y to be datalog queries. This allows specifically to consider relations between objects, and thus enhances the expressiveness of the association rules which can be discovered. The resulting *relational association rules,* however, suffer to an even larger extent the main problem of classical association rule mining: even for a small dataset, the number of resulting rules is very high, and there are many uninteresting and even redundant rules in the result.

In the classical scenario, several solutions (for instance measures of "usefulness" [7]) have been proposed. These approaches can also be applied to relational association rules, but they all have in common that they lose some information.

A complementary approach is based on *Formal Concept Analysis (FCA)* [37,15]. Its simplest data structure, a 'formal context', fits exactly the scenario of classical association

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 109–125, 2004.

rule mining. It turned out that the concepts of the concept lattice provide a condensed representation for frequent itemsets.

This observation was independently made by three research groups around 1997/98: L. Lakhal and his database group in Clermont–Ferrand [22], M. Zaki in Troy, NY [39], and the author in Darmstadt [31]. The first algorithm based on this idea was Close [24], followed by A-Close [23], ChARM [40], PASCAL [5], Closet [26], TITANIC [33,35], and others, each having its own way to exploit the closure system which is hidden in the data. Then different researchers started working on 'condensed representations' of frequent itemsets and association rules: closed sets, free sets, k-free sets, etc. Some of them were well-known in FCA for quite a while, (e. g., closed sets as concept intents — see for instance [14] — and free sets as minimal generators), others (for instance k-free sets) truly extended the set of condensed representations.

In this paper, we discuss how these representations can be applied to datalog queries and to relational association rules. Our approach is as follows: first we re-formulate the problem of mining relational association rules in terms of Formal Concept Analysis. Then we are able to apply in a straightforward manner the full arsenal of FCA-based condensed representations (closed sets, pseudo-closed sets, free sets) to frequent queries. From them, we define *iceberg query lattices* and show by an example that they are adequate for visualizing relational association rules.

This paper continues our work on iceberg concept lattices presented in [35]. It is organized as follows. After giving an introduction to Datalog and describing the problem of mining relational association rules in Section 2, we recall the basics of mining (classical) association rules with Formal Concept Analysis in Section 3. In Section 4 we will restate the relational mining problem in terms of FCA and introduce *iceberg query lattices*. In Section 5, we discuss their use for visualizing relational association rules, before we conclude in Section 6.

2 Relational Association Rules

Relational association rules were inspired by adopting the classical task of mining association rules to ILP (Inductive Logic Programming). First we recall the classical problem, before introducing datalog and relational association rules.

2.1 Association Rules

The problem of mining *association rules* has been discussed for a decade now. It can be stated as follows.

Problem 1 (Association rule mining [1]). Let M be a set of items,[1] G a set of transaction IDs, and I a binary relation between G and M, indicating which items have been bought in which transactions. An *association rule* is a pair $X \to Y$ of subsets of M with

[1] In the scenario of warehouse basket analysis, the items are products sold by the supermarket.

$X \subseteq Y$.[2] Its *support* is the relative frequency of the transactions containing Y among all transactions, and its *confidence* is the relative frequency of all transactions containing Y among those containing X.

The problem of mining association rules is to compute, for given thresholds *minsupp* and *minconf* $\in [0, 1]$, all association rules such that their support and confidence are above (or equal to) the thresholds *minsupp* and *minconf,* resp.

In a supermarket database, for instance, the rule 'salmon → white_wine (conf = 73 %, supp = 3 %)' would indicate that 73 % of all customers buying salmon also buy white wine, and that this combination is bought in 3 % of all transactions.

The standard approach to the problem is to compute first all frequent itemsets Y (i. e., all itemsets with a support above the threshold *minsupp*), and then check for each of them and for each of its subsets X if $\mathit{conf}(X \rightarrow Y) \geq \mathit{minconf}$. The first step is the expensive one, as it requires (multiple) access(es) to the database. Therefore, most research in the domain focuses on this first step.

Relational Association Rules have been introduced by L. Dehaspe and H. Toivonen in [11], following their work on the discovery of frequent datalog patterns [10]. Relational association rules are defined within the framework of *datalog*. This enhances thus the expressiveness of the rules that can be discovered. The basic idea is to replace G and I by a datalog database, and M by a set of datalog queries. Before describing this idea in detail, we recall some basic datalog definitions.

2.2 Datalog

A deductive *datalog database* **r** is a set of definite clauses, i. e., of universally quantified formulas of the form $\forall \boldsymbol{x}. l_o \leftarrow (l_1 \wedge \ldots \wedge l_n)$ with l_i being positive literals, $n \geq 0$, and $\boldsymbol{x}$ consisting of all variables appearing in the literals l_i. We abbreviate the clauses by $l_o \leftarrow l_1, \ldots, l_n$. A clause with $n = 0$ which does not contain any variables is called a *fact*.

A *substitution* θ is a set $\{X_1/a_1, \ldots, X_m/a_m\}$ of bindings of variables X_i to terms a_i. The *instance* $C\theta$ of a clause C for a substitution θ is obtained from C by replacing all occurrences of the variables X_i by the terms a_i, resp. If $C\theta$ is ground (i. e., if it contains only constants as terms, no variables), then it is called *ground instance* of C, and θ is a *grounding substitution.*

A *datalog query* Q is a logical expression of the form $? - l_1, \ldots, l_n$ with $n \geq 1$, where the l_i are *literals* (i. e., predicates or negated predicates). A query Q *succeeds* for a database **r** if there exists a grounding substitution θ for Q such that the conjunction $(l_1 \wedge \ldots \wedge l_n)\theta$ holds within the database, θ is then called *answering substitution for Q*. The set of all answering substitutions for Q in **r** is denoted by *answerset(Q,* **r**).

In order to avoid some logical problems related to negation, we restrict ourselves to *range-restricted* queries, i. e., to queries where all variables that occur in negative literals also occur in at least one positive literal.

[2] Usually, one requires head and body of the rules to be disjoint, but both statements are equivalent, since $X \rightarrow Y$ and $X \rightarrow Y \setminus X$ have the same support and the same confidence. Our version of the problem statement is both closer to the way association rules are computed and more adequate to the problem statement for relational association rules.

Table 1. Example datalog database **r**

customer(*allen*)	parent(*allen*, *bill*)	buys (*allen*, *wine*)
customer(*bill*)	parent(*allen*, *carol*)	buys (*bill*, *cola*)
customer(*carol*)	parent(*bill*, *zoe*)	buys (*bill*, *pizza*)
customer(*diana*)	parent(*carol*, *diana*)	buys (*diana*, *pizza*)

Example 1. We illustrate these concepts using an example from [11], which we will use as running example throughout the paper. The datalog database is shown in Table 1. It consists of facts only. It could be (and usually is) extended by so-called intensional relations as, e. g., $\textit{grandparent}(X, Z) \leftarrow \textit{parent}(X, Y), \textit{parent}(Y, Z)$. which provides an intensional definition of the predicate *grandparent.*

Consider now the query

$$Q \ := \ ?- \textit{customer}(X), \textit{parent}(X, Y), \textit{buys}(Y, \textit{pizza}).$$

Applied to the database **r**, it will return all couples (a_1, a_2) of instances such that the three facts $\textit{customer}(a_1), \textit{parent}(a_1, a_2)$, and $\textit{buys}(a_2, \textit{pizza})$ are all in the database. The result is thus *answerset(Q,* **r**) := {(*X/allen, Y/bill*), (*X/carol, Y/diana*}.

2.3 Relational Association Rules

Although the intuition of 'datalog association rules' is quite straightforward, there are some subtleties to be resolved. For instance, it is not clear from the start what exactly is to be counted. These aspects have been discussed in [11] and led to the following definition of the problem.

Definition: A *relational association rule* (or *query extension*) is an implication of the form

$$?- l_1, \ldots, l_m. \ \rightarrow \ ?- l_1, \ldots, l_m, l_{m+1}, \ldots, l_n.$$

with $1 \leq m < n$, where both parts separately are existentially quantified. The rule may be abbreviated by

$$?- l_1, \ldots, l_m \rightsquigarrow l_{m+1}, \ldots, l_n.$$

$?- l_1, \ldots, l_m.$ is the *body* of the rule, and the subquery $l_{m+1}, \ldots, l_n$ is the *head* of the rule. The *conclusion* of the rule is $?- l_1, \ldots, l_m, l_{m+1}, \ldots, l_n$. We will sometimes write the rule in the form $Q_1 \rightarrow Q_2$, where Q_1 stands for the body of the rule, and Q_2 for the conclusion of the rule.

Note that relational association rules consist of two queries, where the second one extends the first. In the sequel, we will consider the conclusion of the rule as query rather than the head, as only this guarantees that the scope of the existential quantifier is as it is intended. Different ways of 'misinterpreting' this have been discussed in [11].

In the standard case of association rule mining, transactions are counted to define support and confidence. The transaction id is a key in the database (modeled as set *G* in Problem 1), so that counting becomes unambiguous. For relational association rules, this has to be simulated. In [11], one of the relations of the datalog database (called *key*) is taking over this role.

Definition: Let **r** be a datalog database and Q be a query containing an atom *key*. Then the *support* (or *relative frequency*) of query Q with respect to database **r** given *key* is

$$supp(Q, \mathbf{r}, key) := \frac{|\{\theta \in answerset(?-key, \mathbf{r}) \mid Q\theta \text{ succeeds w.r.t. } \mathbf{r}\}|}{|answerset(?-key, \mathbf{r})|} .$$

The *support* of a relational association rule is given by the support of the conclusion of the rule. Its confidence is the support of the conclusion of the rule divided by the support of the body of the rule.

Now we are able to formally state the problem of mining all frequent relational association rules.

Problem 2 (Relational Association Rule Mining [11]). Let **r** be a datalog database and $\mathcal{L}$ a set of datalog queries that all contain an atom *key*. Let *minsupp* and *minconf* be two (user-defined) thresholds between 0 and 1. The task is then to discover among the relational association rules which can be built from $\mathcal{L}$ all rules with support and confidence above the given thresholds.

Lemma 1. *The standard problem of mining association rules is a special case of this scenario.*

Proof: Let M be a set of items, G a set of transaction IDs, and $I \subseteq G \times M$. We introduce a unary predicate *key* which holds for all $g \in G$. We consider each item $m \in M$ as a constant, and introduce a binary predicate $contains(g, m)$ that holds whenever transaction g contains item m. If the set $\mathcal{L}$ contains all relational queries composed of the literal *key(X)* and any combination of literals of the form $contains(X, m)$, then mining all relational association rules is equivalent to mining all association rules in the classical sense. □

Example 2. In our running example, we are able to discover for instance the following rules. In brackets are shown support (as decimal number) and confidence (as fraction). They are based on the *customer* predicate as key.

$$? - customer(A), buys(A, wine) \rightsquigarrow parent(A, B), parent(B, C) \qquad (0.25, 1/1)$$
$$? - customer(A), parent(A, B) \rightsquigarrow buys(B, cola) \qquad (0.25, 1/3)$$
$$? - customer(A), parent(A, B) \rightsquigarrow buys(B, cola), buys(B, pizza) \qquad (0.25, 1/3)$$

The first rule states that each customer buying wine is also a grandparent. The second rule states that a third of all customers having a child also have a child buying cola. (Remark that this statement is different from the following: a third of all children is buying cola. In the first case, the parents are counted, and in the second the children.) The last statement says that a third of all customers having a child also have a child buying cola and pizza.

The declarative language bias used in this example restricts the construction of queries in the following way:[3] Variable *A* is bound by the *customer* predicate, and may serve as input in the first position of *parent* and/or *buys.* The *parent* predicate may be iterated, while *buys* is forced to have one of the constants *cola, pizza,* or *wine* at the second position. This prohibits for instance queries like 'return all customers buying the same item as one of their children'. This bias allows to fine-tune the trade-off between the size of the set of rules and the expressive power of the rules.

As in the classical case, this problem is naturally split in two sub-problems: first compute all frequent queries $?- l_1, \ldots, l_n.$, and then check the support of all rules of the form $?- l_1, \ldots, l_m \rightsquigarrow l_{m+1}, \ldots, l_n.$ by considering all possible partitions of the set of literals. In this paper, we focus on the first step.

In [11], an algorithm for computing frequent queries, called WARMR, is presented, which basically is a first order variant of the well known Apriori algorithm [2]. In order to reduce the resulting rules to a set of 'useful rules', WARMR makes additional use of a *declarative language bias* as known from Inductive Logic Programming (ILP). The basic idea is to fix the order in which the variables are bound. Details can be found in[11].

As in the case of classic association rule mining, a major problem for mining relational association rules is the high number of resulting rules. (In fact, the problem is much larger in the new scenario.) In the classical case, a number of additional measures for 'interestingness' have been introduced to reduce further the number of rules (see for instance [7]). These measures can of course also be applied to relational association rules. In [11], Dehaspe and Toivonen additionally discuss the statistical significance of the confidence, and the declarative language bias discussed above to further reduce the number of rules. All of these approaches reduce the number of rules, but for the price of loosing some information.

In the next section, we will discuss how quite a number of frequent queries — and subsequently of rules — can be pruned *without* loosing any information. Only if our pruning does not sufficiently reduce the number of rules, then additional means are needed to continue pruning (with loss of information).

The basic idea of our approach is based on the observation that some rules talk about exactly the same set of instances (and thus with exactly the same support and the same confidence), but on different levels of detail. In that case, the more specific rule can be pruned without loss of information. In Example 2, for instance, the second rule is comprised by the third, and both rules are talking about exactly the same instance, namely *allen.* Hence it is sufficient to present the third rule to the user; the second can be pruned without loosing any information.[4] In the next section, we discuss the theoretical background for this kind of pruning.

[3] In WARMODE format, this is stated as "warmode_key(customer(-)). warmode(parent(+,-)). warmode(buys(+, cola)). warmode(buys(+,pizza)). warmode(buys(+,wine))." '-' indicates that the variable is bound by the atom, and '+' means that the variable is bound before the atom is called.

[4] More precisely, the second rule can derived from the third rule together with the exact rules discussed in Section 5.

3 Formal Concepts and Association Rules

Consider again the classical problem of mining association rales (Problem 1). Assume that there are two itemsets which both describe exactly the same set of transactions. So if we know the support of one of them, we do not need to count the support of the other one in the database. In fact, we can introduce an equivalence relation Ψ on the set of itemsets. Two itemsets are said to be *equivalent with respect to a database* if and only if they are contained in exactly the same set of transactions. If we knew the relation from the beginning, it would be sufficient to count the support of one itemset of each class only — all other supports can then be derived.

Of course one does not know the equivalence relation in advance, but one can determine it along the computation. It turns out that one usually has to count the support of more than one query of each class, but normally not of all of them. The percentage of queries to be considered depends on how correlated the data are: the more correlated the data are, the fewer counts have to be performed.

3.1 Formal Concept Analysis

Condensed representations of frequent itemsets (e. g., free or closed sets) were inspired by the theory of Formal Concept Analyis [22,39,31]. We assume the reader to be familiar with the basic notions of *Formal Concept Analysis (FCA)* and refer to [15] otherwise. For keeping notations consistent, however, we recall the most important definitions.

Definition: A *(formal) context* $\mathbb{K} := (G, M, I)$ consists of a set G of objects, a set M of attributes, and a binary relation $I \subseteq G \times M$.

The mapping $\cdot^{\uparrow}\colon \mathfrak{P}(G) \to \mathfrak{P}(M)$ is given by $A^{\uparrow} := \{m \in M \mid \forall g \in A\colon (g, m) \in I\}$. The mapping $\cdot^{\downarrow}\colon \mathfrak{P}(M) \to \mathfrak{P}(G)$ is defined dually by $B^{\downarrow} := \{g \in G \mid \forall m \in B\colon (g, m) \in I\}$. If it is clear from the context whether the first or the second mapping is meant, then we abbreviate both $\cdot^{\uparrow}$ and $\cdot^{\downarrow}$ just by $\cdot'$. In particular, B'' stands for $(B^{\downarrow})^{\uparrow}$.

A *(formal) concept* is a pair *(A, B)* with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called *extent* and B is called *intent* of the concept. The set $\underline{\mathfrak{B}}(\mathbb{K})$ of all concepts of a formal context $\mathbb{K}$ together with the partial order $(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2$ (which is equivalent to $B_1 \supseteq B_2$) is called *concept lattice* of $\mathbb{K}$.

3.2 Mining Association Rules with FCA

Obviously, a formal context is just the right data structure for describing the problem of classical association rule mining (Problem 1), and FCA provides indeed a natural framework for the task of mining association rules. The equivalence relation Ψ mentioned above is formalized as follows: for $X, Y \in \mathfrak{P}(M)$, let $X\Psi Y$ if and only if $X' = Y'$. It turns out that each concept intent is exactly the largest itemset of its equivalence class. The concept intents are also called *closed itemsets,* as they are exactly the closed sets of the canonical closure operator $B \mapsto B''$ on $\mathfrak{P}(M)$. For any itemset $B \in \mathfrak{P}(M)$, its closure is the set B'', which is just the concept intent of its equivalence class. Note that in particular holds, for any $\hat{B} \in \mathfrak{P}(M)$ with $B\Psi\hat{B}$, $supp(\hat{B}) = supp(B) = supp(B'')$.

The (frequent) concept intents/closed itemsets can hence be considered as 'normal forms' of the (frequent) itemsets.

The *support* of a formal concept *(A, B)* of $\mathbb{K}$ is the support of the itemset B (which is equal to the ratio $\frac{|A|}{|G|}$). The concepts with a support above or equal to a user-defined threshold *minsupp* $\in$ [0,1] are called *frequent concepts,* and the set of all frequent concepts is called *iceberg concept lattice* [35]. While it is not of particular interest to study the set of all frequent (closed and non-closed) itemsets, the iceberg lattice provides interesting insights into the data. In [4,34,25], we showed how the number of association rules can be reduced by using the iceberg concept lattice, and how the latter can be used for visualizing the rules.

In particular, the iceberg concept lattice contains all information needed to derive the support of all (frequent) itemsets. This observation has been exploited first in the Close algorithm [22] to improve the efficiency of algorithms for mining frequent itemsets. Instead of using the maximal elements of the equivalence classes, one can also use their *minimal generators* (which are now called *free sets* or *key sets* in data mining) [4, 34]. They can be computed, together with the closed sets, e.g., with the algorithm TITANIC [35]. Rather than recalling these results here, we directly apply them to the task of relational association rule mining.

4 Iceberg Concept Lattices for Datalog Queries

Let us now come back to the problem of mining frequent datalog queries.

Example 3. Figure 1 shows all 32 queries which follow the declarative language bias introduced above, and which have at least one answering substitution.[5] Each node in the diagram stands for one query, which consists of all literals that are attached at the node itself, or at some node above (following straight lines only). A query succeeds in our example database with all answering substitutions which are attached to the node of the query itself or to any node below in the hierarchy. If parts of the query are logically redundant, then only the relevant part of the answering substitution is given. The numbers in the nodes indicate the support as discussed in Section 4.2. For the moment, we just ignore them.

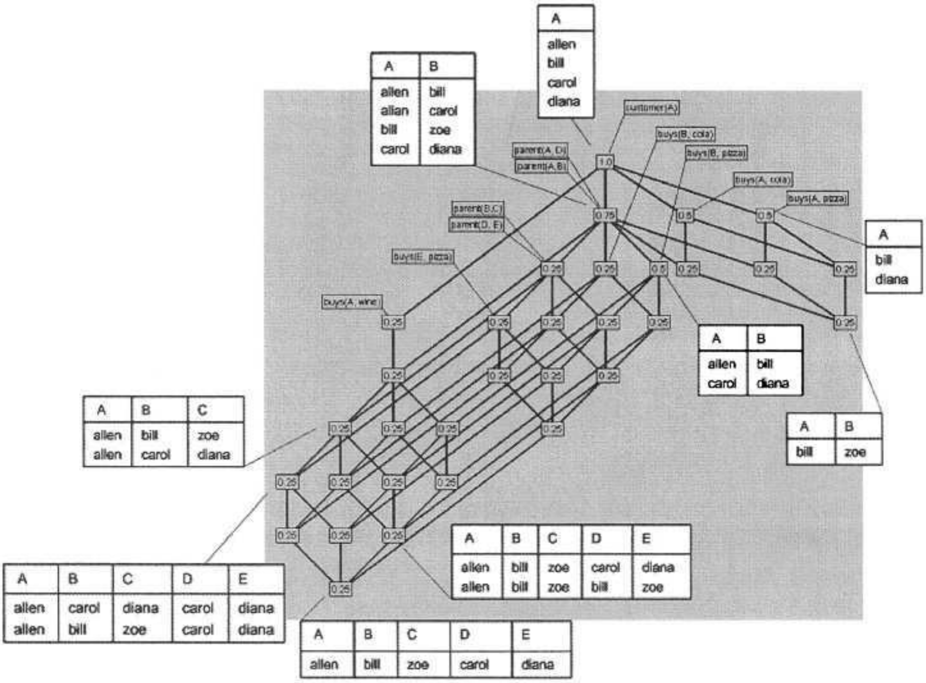
For instance, the right-most lower node stands for the query

$$?-\ \boldsymbol{customer}(A), \boldsymbol{parent}(A,B), \boldsymbol{parent}(A,D), \boldsymbol{buys}(A,\boldsymbol{cola}), \boldsymbol{buys}(A,\boldsymbol{pizza}).$$

In this case, the literal *parent(A, D)* is logically redundant, and may be removed. It is needed on the left side of the diagram, though, where we have to distinguish between the different grandchildren of *allen.*

Note that we do not talk about sets of queries (as we would talk about sets of items in the classical case), but only about single queries. This stems from the fact that

[5] Remark that [11] lists only 26 of these frequent queries; all queries where both *parent(A, B)* and *parent(A, D)* are required are omitted. The reason seems to be that WARMR prunes logically redundant queries immediately when it passes them, even though they may be needed for building up more specific queries, as, e.g., the query represented by the left-most lower node.



A
allen bill carol diana

A	B
allen	bill
allan	carol
bill	zoe
carol	diana

A
bill diana

A	B
allen	bill
carol	diana

A	B
bill	zoe

A	B	C
allen	bill	zoe
allen	carol	diana

A	B	C	D	E
allen	bill	zoe	carol	diana
allen	bill	zoe	bill	zoe

A	B	C	D	E
allen	carol	diana	carol	diana
allen	bill	zoe	carol	diana

A	B	C	D	E
allen	bill	zoe	carol	diana

Fig. 1. All queries following the declarative language bias which have at least one answering substitution.

(unlike in the classical situation where the combination of items is not an item itself) the combination of queries is again a query, since the set of datalog queries is closed under conjunction. We can hence *identify* any finite set of queries with the conjunction of its queries.[6] Therefore, we assume in the sequel that the set $\mathcal{L}$ of datalog queries that we consider is always closed under conjunction.

While it is not really informative to study the set of all frequent queries, the situation changes when we consider the *closed queries* among them only. In order to define them formally, we first have to bring together the notions of datalog and FCA. We will consider two formal contexts that canonically arise from our scenario. Both definitions give rise to different understandings of 'closed queries'. They are discussed in the two following subsections, resp.

[6] Eventually variable renaming has to be performed (in the usual first order logic way) before the conjunction is computed; in order to respect the range of the existential quantifiers of the individual queries.

4.1 Iceberg Query Lattices of Datalog Databases

Definition: Let $\mathbf{r}$ be a datalog database and $\mathcal{L}$ a set of datalog queries. The *formal context associated to* $\mathbf{r}$ *and* $\mathcal{L}$ is defined by $\mathbb{K}_{\mathbf{r},\mathcal{L}} := (G_{\mathbf{r},\mathcal{L}}, M_{\mathbf{r},\mathcal{L}}, I_{\mathbf{r},\mathcal{L}})$ where $G_{\mathbf{r},\mathcal{L}} := \{\theta \mid \theta$ is a grounding substitution for all $L \in \mathcal{L}\}$, $M_{\mathbf{r},\mathcal{L}} := \mathcal{L}$, and $(\theta, Q) \in I_{\mathbf{r},\mathcal{L}}$ if and only if $\theta \in \mathit{answerset}(Q, \mathbf{r})$.

From this formal context, one can compute the concept lattice as usual. As discussed above, we may identify the intent of a formal concept (A, B) of this lattice with the query $\bigwedge B$, which stands for the conjunction of all queries contained in B. Such a query is also called *closed query with respect to* $\mathbf{r}$ *and* $\mathcal{L}$, as it is related to the closed set B. Like in the classical scenario, one can introduce an equivalence relation $\Psi_{\mathbf{r}}$ on the set of queries. Two queries Q_1 and Q_2 are said to be *equivalent with respect to database* $\mathbf{r}$ if and only if $\mathit{answerset}(Q_1, \mathbf{r}) = \mathit{answerset}(Q_2, \mathbf{r})$. The most specific query of each equivalence class is then just the closed query which is assigned to the corresponding formal concept of the context $\mathbb{K}_{\mathbf{r},\mathcal{L}}$.

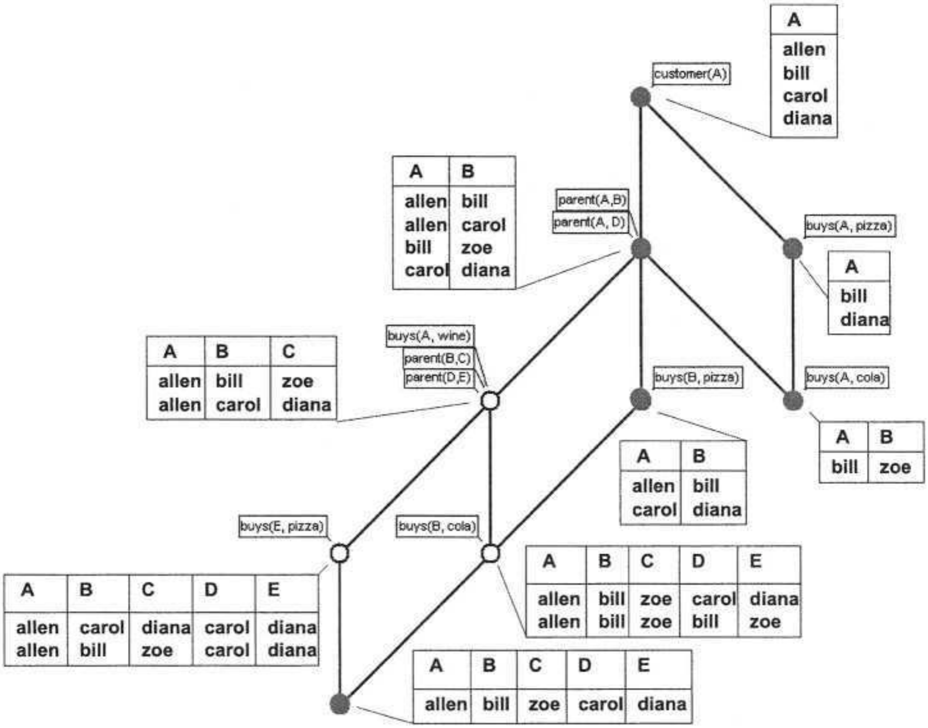
Definition: Let $\mathbf{r}$ be a datalog database and $\mathcal{L}$ a set of datalog queries. The *iceberg query lattice of* $\mathbf{r}$ *and* $\mathcal{L}$ for *minsupp* $\in [0, 1]$ is given by $\underline{\mathfrak{B}}(\mathbf{r}, \mathcal{L}) := (\{Q \in \mathcal{L} \mid Q$ is closed with respect to $\mathbf{r}$ and $\mathcal{L}$, and the corresponding concept is frequent$\}, \models)$, where $\models$ is the usual logical implication.

Example 4. Figure 2 shows all frequent closed queries of our running example, where 'frequent' means support strictly larger than 0. The diagram is read in the same way as in the previous figure: a node represents the query which consists of all literals labeled at it and at all higher nodes. As in Figure 1, the diagram shows the relevant parts of the answering substitutions for each query. The bold nodes are discussed in the next subsection.

In Figure 1, the nine frequent closed queries are exactly those which are labeled by an answering substitution. Each of the 32 frequent queries belongs to the same equivalence class of $\Psi_{\mathbf{r}}$ as the highest closed query which is below it (i. e., to its most general closed specialization). The right-most upper query ?– *customer(A)*, *buys(A, cola), buys(A, pizza).*, for instance, is in the same class as the closed query which is just below it (and which has as additional literal *parent(A, B)*).

Without any loss of information, the diagram in Figure 2 gives a much better insight into the database. It shows for instance that being grandparent and buying wine is equivalent in our example, since *buys(A, wine)* and *parent(B, C)* generate the same node. It also shows that any *customer* buying *cola* also buys *pizza* and is *parent* of someone. This *implication* (or *exact association rule*) is indicated by the fact that the node labeled by *buys(A, cola)* is below the nodes labeled by *parent(A, B)* and *buys(A, pizza)*, resp., in the diagram. This is the general way implications are read in concept lattices.

It is obvious that the restriction to frequent closed queries gives a much better insight into the content of the database. One drawback, however, — at least for the association rule scenario — still exists: the meaning of counting objects is not intuitively clear. As the size of the 'relevant part' of an answering substitution depends on the number of variables involved, it is not clear what exactly has to be counted. If one requires the user



A
allen
bill
carol
diana

A	B
allen	bill
allen	carol
bill	zoe
carol	diana

A
bill
diana

A	B	C
allen	bill	zoe
allen	carol	diana

A	B
allen	bill
carol	diana

A	B
bill	zoe

A	B	C	D	E
allen	carol	diana	carol	diana
allen	bill	zoe	carol	diana

A	B	C	D	E
allen	bill	zoe	carol	diana
allen	bill	zoe	bill	zoe

A	B	C	D	E
allen	bill	zoe	carol	diana

Fig. 2. All frequent closed queries.

to provide meaningful values for *minsupp* and *minconf* for the mining task, then this question has to be answered. That is the reason why Dehaspe and Toivonen introduced an explicit *key* predicate in [11]. We discuss their approach in the light of FCA next.

4.2 Iceberg Query Lattices of Datalog Databases with Respect to a Key Predicate

Again, we first transform the datalog database into a formal context. The difference to the approach discussed above is that we now consider only the instances of the *key* predicate as objects.

Definition: Let $\mathbf{r}$ be a datalog database and $\mathcal{L}$ be a set of datalog queries which all contain an atom *key*. The *formal context associated to* $\mathbf{r}$, $\mathcal{L}$, *and key* is defined by $\mathbb{K}_{\mathbf{r},\mathcal{L},key} := (G_{\mathbf{r},\mathcal{L},key}, M_{\mathbf{r},\mathcal{L},key}, I_{\mathbf{r},\mathcal{L},key})$ where $G_{\mathbf{r},\mathcal{L},key} := answerset(?- key, \mathbf{r})$, $M_{\mathbf{r},\mathcal{L},key} := \mathcal{L}$, and $(\theta, Q) \in I_{\mathbf{r},\mathcal{L},key}$ if and only if there exists a grounding substitution $\hat{\theta}$ for Q with $\theta \subseteq \hat{\theta}$.

Two queries $Q_1, Q_2 \in \mathcal{L}$ are *equivalent with respect to* $\mathbf{r}$ *and key* (denoted $Q_1 \Psi_{\mathbf{r},key} Q_2$) if $Q_1' = Q_2'$ holds in $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$. *Closed queries of* $\mathbf{r}$ *and* $\mathcal{L}$ *with respect to key and minsupp* $\in$ [0,1] are defined as above. The *iceberg query lattice* of $\mathbf{r}$, $\mathcal{L}$,

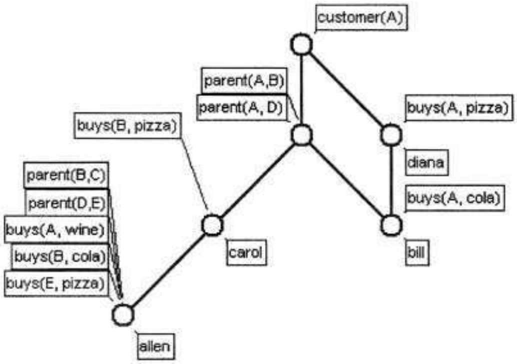


Fig. 3. All frequent closed queries for the *key* predicate *'customer'*.

and *key* for *minsupp* is $\underline{\mathfrak{B}}(\mathbf{r}, \mathcal{L}, key) := (\{Q \in \mathcal{L} \mid Q$ is closed and $supp(Q, \mathbf{r}, key) \geq minsupp\}, \models)$.

Example 5. Figure 3 shows all six queries which are frequent according to this definition for any *minsupp* $\in$ (0, 0.25]. The closed queries displayed here are also closed queries of the context $\mathbb{K}_{\mathbf{r},\mathcal{L}}$. In Figure 2, they are the ones marked by filled nodes. Theorem 1 below shows that this containment holds in general.

As in the previous example, we can read off implications between queries from the diagram. In particular, we rediscover the implications discussed in Figure 4: being grandparent and buying wine is equivalent; and any *customer* buying *cola* also buys *pizza* and is *parent* of someone. But as we have now a coarser look to the data, there are more equivalences to be discovered in this representation. Indeed, we focus on the *customers,* and do not distinguish explicitly between the different family lines of customer *allen* any more: the diagram shows that having a grandchild buying *pizza* is equivalent to having a child buying *cola* (who needs not be *parent* of the grandchild). So by choosing which of the contexts $\mathbb{K}_{\mathbf{r},\mathcal{L}}$ and $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$ to study, we can decide how close to look at the relations between the instances.

The numbers in Figure 1 show the support of each query measured in the context $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$. If a query Q_1 logically subsumes another query Q_2 (i.e., Q_1 is below Q_2 in the diagram) and $supp(Q_1) = supp(Q_2)$ holds, then both queries have the same closure (and are in the same class of $\Psi_{\mathbf{r},key}$). As the closed queries are the most specific in their equivalence classes, they are exactly those queries whose support is different from all supports of the queries which are immediately below it (see [35] for details). In Figure 1, the six queries which are closed with respect to *key* predicate *'customer'* are thus: the top-most query, the one immediately below it, the queries labeled by *buys(A,pizza), buys(B,pizza),* and the two queries which do not have any lower neighbors.

The following theorem shows the general relationship between the (iceberg) query lattices of the formal contexts introduced in this and in the previous subsection.

Theorem 1. *Let* **r** *be a datalog database and* $\mathcal{L}$ *be a set of datalog queries closed under conjunction where all queries contain an atom key.* $\mathfrak{B}(\mathbf{r}, \mathcal{L}, key)$ *is a* $\vee$***-sub-semi-lattice*** *of* $\mathfrak{B}(\mathbf{r}, \mathcal{L})$. *Here, '*$\vee$*' can be read both as join (supremum) in the concept lattice or as operator returning the most specific common generalization of two queries.*

Proof: For $\mathbb{K}_{\mathbf{r},\mathcal{L}}$ and $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$, we have $M_{\mathbf{r},\mathcal{L}} = M_{\mathbf{r},\mathcal{L},key}$. We show that for each $\theta \in G_{\mathbf{r},\mathcal{L},key}$ exists a $\hat{\theta} \in G_{\mathbf{r},\mathcal{L}}$ with $\{\theta\}' = \{\hat{\theta}\}'$ where $\cdot'$ is computed in the corresponding context, resp. This proves the theorem by Lemma 31 of [15].

Let $\theta \in G_{\mathbf{r},\mathcal{L},key}$, and let Q be the most specific query in $\mathcal{L}$ returning θ for $key(\boldsymbol{x}) \leftarrow Q(\boldsymbol{x})$. Q exists since $\mathcal{L}$ is closed under conjunction. Let $\hat{\theta}$ be the answering substitution for Q. Then $\{\theta\}' = \{\hat{\theta}\}'$ holds. □

We conclude this section by showing that the definition of the formal context associated to **r**, $\mathcal{L}$, and *key* is the right way to model Problem 2: this problem is indeed a specific instance of Problem 1 for the context $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$. As the confidence of a rule is always derived from the support of the itemsets/queries involved, it is sufficient to consider the itemsets/queries rather than the (relational) association rules. The proof of the following result is straightforward.

Theorem 2. *Let* **r** *be a datalog database,* $\mathcal{L}$ *be a set of datalog queries that all contain an atom key, and let minsupp be a (user-defined) threshold between 0 and 1. Then the set of frequent queries (in the sense of Problem 2) is equal to the set of frequent items of* $\mathbb{K}_{\mathbf{r},\mathcal{L},key}$ *(in the sense of Problem 1).*

5 Visualizations of Relational Association Rules in Iceberg Query Lattices

In [4,25][7] and [34], we showed how the number of (classical) association rules can be reduced without any loss of information by applying FCA. While the first approach is based on free sets (i.e., the head of a rule is a free set, while the conclusion is a closed set), the second approach is based on closed sets (i.e., both head and conclusion are closed sets). In this paper, we transfer the results of [34] to relational association rules, and show how they can be used for visualizing relational association rules within iceberg query lattices.

We distinguish between two types of rules. *Exact rules* (or *implications*) hold with 100% confidence, while the confidence of *approximate rules* is strictly lower. In the following two subsections, we discuss how the two kinds of rules can be visualized in the same diagram. Because of space restrictions, we can only provide examples here.

5.1 Visualizing the Exact Rules

The visualization of implications in the (iceberg) concept lattice is a powerful tool for communication, which has been used in FCA all along its twenty-five years history.

[7] Similar results have been presented independently in [41].

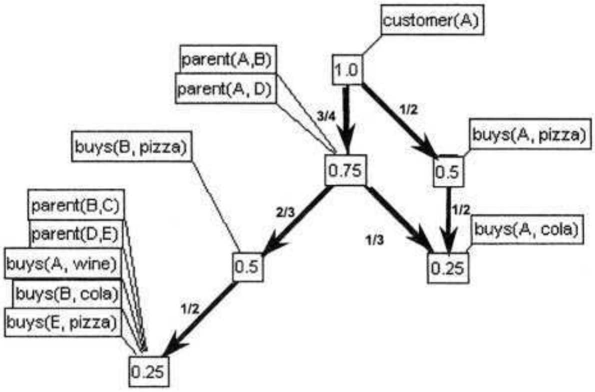


Fig. 4. All frequent relational association rules for the *key* predicate *'customer'*.

It is straightforward to apply it to iceberg query lattices: a relational association rule $?-l_1,\ldots,l_m \rightsquigarrow l_{m+1},\ldots,l_n$. is an exact rule if and only if the largest node which is below all nodes labeled by the literals $l_1,\ldots, l_m$ of the body of the rule is also below all labels $l_{m+1},\ldots, l_n$ of the head of the rule.

Example 6. Consider again Figure 3. The first rule from Example 2 holds with confidence 1/1, since the largest node below the two literals *customer(A)* and *buys(A, wine)* is the node labeled by *allen,* and this node is also below the labels *parent(A, B)* and *parent(B, C).* Similarly, the rule

$$?-\mathit{customer}(A),\mathit{parent}(A,B),\mathit{buys}(A,\mathit{pizza}) \rightsquigarrow \mathit{buys}(A,\mathit{cola})$$

holds, since the largest node below *customer(A), parent(A, B),* and *buys(A,pizza)* is the one labeled by *bill,* and this node is also below (more precisely: at) the label *buys(A, cola).* There is no frequent exact rule having

$$?-\mathit{customer}(A),\mathit{parent}(A,B),\mathit{buys}(B,\mathit{pizza}),\mathit{buys}(A,\mathit{cola})$$

as head, as there is no node below these literals.

5.2 Visualizing the Approximate Rules

In [34], we show that it is sufficient to consider only rules $Q_1 \rightarrow Q_2$ where both Q_1 and Q_2 are closed and where Q_2 is an immediate specialization of Q_1. From these, all other frequent rules can be derived.

By considering only these specific rules, they all correspond to edges in the line diagram of the iceberg query lattice $\underline{\mathfrak{B}}(\mathbf{r},\mathcal{L},\mathit{key})$. Therefore, we can label each such edge by the confidence of the rule, and each node by the support of the corresponding query. The support of a rule is then the support labeled at the node the rule is pointing to.

Example 7. Figure 4 shows the Luxenburger basis for our running example. The arrow labeled with '2/3', for instance, stands for the rule ? – *customer(A),parent(A, B)* $\leadsto$ *buys(B, pizza).,* which holds with confidence 2/3 and support 0.5. The third rule of Example 2 is given by the arrow labeled with '1/3'.

Rules can also be composed. For instance, the rule ?— *customer(A), parent(A, B)* $\leadsto$ *buys(A, wine).* is composed of the two rules pointing to the left. It has thus confidence $2/3 \cdot 1/2 = 1/3$ and support 0.25.

6 Conclusion

In this paper, we introduced two kinds of iceberg query lattices as different condensed representations of frequent datalog queries. We argued that by switching between them one can decide how close to analyze the relations between instances. We also demonstrated that iceberg query lattices provide a natural way to visualize relational association rules in a non-redundant way.

References

1. R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. *Proc. SIGMOD Conf.,* 1993, 207–216
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rales. *Proc. VLDB Conf,* 1994, 478–499 (Expanded version in IBM Report RJ9839)
3. R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. 11th Intl. Conference on Data Engineering (ICDE '95),* Taipeh, Taiwan, March 1995, 3–14
4. Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, L. Lakhal: Mining minimal non-redundant association rules using frequent closed itemsets. In: J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.–K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, P. J. Stuckey (Eds.): *Computational Logic — CL 2000.* Proc. 1st Intl. Conf. on CL (6th Intl. Conf. on Database Systems). LNAI **1861**, Springer, Heidelberg 2000, 972–986
5. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, L. Lakhal: Mining Frequent Patterns with Counting Inference. *SIGKDD Explorations* **2**(2), Special Issue on Scalable Algorithms, 2000, 66–75
6. R. J. Bayardo. Efficiently mining long patterns from databases. *Proc. SIGMOD Conf,* 1998, 85–93
7. R. J. Bayardo, R. Agrawal, D. Gunopulos. Constraint-based rale mining in large, dense databases. *Proc. ICDE Conf,* 1999, 188–197
8. S. Brin, R. Motwani, J. D. Ullman, S. Tsur: Dynamic itemset counting and implication rules for market basket data. *Proc. SIGMOD Conf.,* 1997, 255–264
9. C. Carpineto, G. Romano: GALOIS: An Order-Theoretic Approach to Conceptual Clustering. *Machine Learning.* Proc. ICML 1993, Morgan Kaufmann Prublishers 1993, 33–40
10. L. Dehaspe, H. Toivonen: Discovery offrequent datalog patterns. *Data Mining and Knowledge Discovery* **3**, 1999, 7–36
11. L. Dehaspe, H. Toivonen: Discovery of Relational Association Rules. In: S. Džeroski, N. Lavrač (Eds.): *Relational Data Mining.* Springer, Heidelberg 2001, 189–212
12. H. Dicky, C. Dony, M. Huchard, T Libourel: On automatic class insertion with overloading. *OOPSLA 1996,* 251–267

13. U. M. Fayyad, G. Piatetsky–Shapiro, P. Smyth, R. Uthurusamy (eds.): Advances in knowledge discovery and data mining. AAAI Press, Cambridge 1996
14. B. Ganter, K. Reuter: Finding all closed sets: A general approach. *Order.* Kluwer Academic Publishers, 1991, 283–290
15. B. Ganter, R. Wille: *Formal Concept Analysis: Mathematical Foundations.* Springer, Heidelberg 1999
16. R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, T. Chau: Design of class hierarchies based on concept (Galois) lattices. *TAPOS* **4**(2), 1998, 117–134
17. J.-L. Guigues, V. Duquenne: Famille minimale d'implication informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines* 24(95), 1986, 5–18
18. M. Luxenburger: Implications partielles dans un contexte. *Mathématiques, Informatique et Sciences Humaines,* 29(113), 1991, 35–55
19. M. Luxenburger: Partial implications. Part I of *Implikationen, Abhängigkeiten und Galois Abbildungen.* PhD thesis, TU Darmstadt. Shaker, Aachen 1993
20. G. Mineau, G., R. Godin: Automatic Structuring of Knowledge Bases by Conceptual Clustering. *IEEE Transactions on Knowledge and Data Engineering* **7**(5), 1995, 824–829
21. M. Missikoff, M. Scholl: An algorithm for insertion into a lattice: application to type classification. *Proc. 3rd Intl. Conf. FODO 1989.* LNCS **367**, Springer, Heidelberg 1989, 64–82
22. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Pruning closed itemset lattices for association rules. *Proc. 14ièmes Journées Bases de Données Avancées (BDA '98),* Hammamet, Tunisie, 177–196
23. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Discovering frequent closed itemsets for association rules. *Proc. ICDT Conf.,* 1999, 398–416
24. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal: Efficient mining of association rules using closed itemset lattices. *Journal of Information Systems,* 24(1), 1999, 25–46
25. N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, L. Lakhal: Generating a Condensed Representation for Association Rules. *J. Intelligent Information Systems (JIIS)* (accepted)
26. J. Pei, J. Han, R. Mao: CLOSET: An efficient algorithm for mining frequent closed itemsets. *Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000,* 21–30
27. I. Schmitt,G. Saake: Merging inheritance hierarchies for database integration. *Proc. 3rd IFCIS Intl. Conf. on Cooperative Information Systems,* New York City, Nework, USA, August 20-22, 1998, 122–131
28. R. Srikant, R. Agrawal: Mining generalized association rules. *Proc. VLDB Conf.,* 1995, 407–419
29. R. Srikant, Q. Vu, R. Agrawal: Mining association rules with item constraints. *Proc. KDD Conf.,* 1997, 67–73
30. S. Strahringer, R. Wille: Conceptual clustering via convex-ordinal structures. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and Classification.* Springer, Berlin-Heidelberg 1993, 85–98
31. G. Stumme: *Conceptual Knowledge Discovery with Frequent Concept Lattices.* FB4-Preprint **2043**, TU Darmstadt 1999
32. G. Stumme: Off to New Shores — Conceptual Knowledge Discovery and Processing. *Intl. J. Human–Comuter Studies (IJHCS)* **59**(3), September 2003, 287–325
33. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Fast Computation of Concept Lattices Using Data Mining Techniques. *Proc. 7th Intl. Workshop on Knowledge Representation Meets Databases,* Berlin, 21–22. August 2000. CEUR-Workshop Proceeding. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/

34. G. Stumme, R. Taouil, Y. Bastide. N. Pasquier, L. Lakhal: Intelligent Structuring and Reducing of Association Rules with Formal Concept Analysis. In: F. Baader. G. Brewker, T. Eiter (Eds.): *KI 2001: Advances in Artificial Intelligence.* Proc. KI 2001. LNAI **2174,** Springer, Heidelberg 2001, 335–350

35. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, L. Lakhal: Computing Iceberg Concept Lattices with Titanic. *J. on Knowledge and Data Engineering (KDE)* **42**(2), 2002, 189–222

36. K. Waiyamai, R. Taouil, L. Lakhal: Towards an object database approach for managing concept lattices. *Proc. 16th Intl. Conf. on Conceptual Modeling,* LNCS **1331,** Springer, Heidelberg 1997, 299–312

37. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.). *Ordered sets.* Reidel, Dordrecht–Boston 1982, 445–470

38. A. Yahia, L. Lakhal, J. P. Bordat, R. Cicchetti: iO2: An algorithmic method for building inheritance graphs in object database design. *Proc. 15th Intl. Conf. on Conceptual Modeling.* LNCS **1157**, Springer, Heidelberg 1996, 422–437

39. M.J. Zaki, M. Ogihara: Theoretical Foundations of Association Rules, *3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD),* Seattle, WA, June 1998, 7:1–7:8

40. M. J. Zaki, C.–J. Hsiao: ChARM: An efficient algorithm for closed association rule mining. Technical Report 99–10, Computer Science Dept., Rensselaer Polytechnic Institute, October 1999

41. M. J. Zaki: Generating non-redundant association rules. *Proc. KDD 2000.* 34–43

‘Particles’ and ‘Waves’ as Understood by Temporal Concept Analysis

Karl Erich Wolff

Mathematics and Science Faculty
Darmstadt University of Applied Sciences
Schoefferstr. 3, D-64295 Darmstadt, Germany
karl.erich.wolff@t-online.de
http://www.fbmn.fh-darmstadt.de/home/wolff

Abstract. A qualitative mathematical framework for a granular representation of relational information about general objects, as for examples particles, waves, persons, families, institutions or other systems in discrete or continuous “space-time” is introduced. That leads to a common description of particles and waves in physics, such that the intuitive physical notion of “wave packet” can be captured by the mathematical notion of a *general object* or a *packet* in a *spatio-temporal Conceptual Semantic System.* In these systems *particles* and *waves* are defined as special packets. It is proved, that “classical objects” and “classical waves” can be represented as, respectively, *particles* and *waves* in spatio-temporal Conceptual Semantic Systems.

The underlying mathematical framework is Formal Concept Analysis and its subdiscipline Temporal Concept Analysis. That allows for describing continuously fine or arbitrarily coarse granularity in the same conceptual framework. The central idea for a “symmetric” description of objects, space, and time is the introduction of “instances” representing information units of observations, connecting the granules for objects, space, and time.

1 Granular Representation of Relational Information

This paper introduces a qualitative mathematical framework for a semantically based granular representation of relational information about general objects, including particles and waves in discrete or continuous “space-time”. In that framework “ semantic scales” are introduced as formal contexts, whose formal concepts are used as values in an ordinal context describing the accepted relational judgments in terms of the concepts of the semantic scales. That allows for describing continuously fine or arbitrarily coarse granularity in the same conceptual framework. Usual space-time theories are not concerned with a theoretical representation of granularity - the success of Euclidean geometry and the precise Newtonian laws led to the conviction that granularity is a good tool for the practitioner, but not for the theorist. Einstein was aware of that problem when he wrote his “granularity” footnote in [4], p. 893:

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 126–141, 2004.

> The inaccuracy which lies in the concept of simultaneity of two events at (about) the same place and which has to be bridged also by an abstraction, shall not be discussed here.

In the following, we discuss the "inaccuracy which lies in the concept of simultaneity of two events" using a granularity notion for "time granules"; we also introduce a granularity notion for "space granules" and "object granules". The mathematical background of Temporal Concept Analysis is described in the next section.

1.1 Temporal Concept Analysis

Temporal Concept Analysis is a theory for describing temporal phenomena with the tools of *Formal Concept Analysis* as introduced by Wille [11,12,5]. We assume that the reader is familiar with the basic notions in Formal Concept Analysis. For an introduction we refer to [12,14].

In Temporal Concept Analysis (TCA) as developed by the author [15,18, 19,21,22] "objects" are studied with respect to "space" and "time". For that purpose the author has introduced the notion of a *Conceptual Time System* which allows for defining mathematically *states, situations* [15], *transitions* [18], and *life tracks* [19,22]. The central structure introduced in [19] is called a "Conceptual Time Systems with Actual Objects and a Time Relation" (CTSOT). The CTSOTs are designed for describing "timeless objects" in the sense, that the definition of a CTSOT starts with a set P whose elements are called objects (or persons or particles). The connection of objects with time is then represented by taking the *actual objects* (p,g), where p is an object and g is a "time granule", as formal objects of a (many-valued) context. A binary *time relation* connecting the actual objects yields a first connection to the "genidentity" in the sense of Reichenbach [7] which can be formally described as a relation R between two actual objects (p,g) and (q,h) defined by $(p,g)\, R\, (q,h) \iff p = q$ and $g < h$, where "<" denotes a strict order relation on the set of time granules. That description of the "genidentity" is not satisfactory since it uses "timeless objects". Therefore, the author was led to a conceptual investigation of "general objects" and their relation to "space" and "time" which allows for defining "objects" as well as "waves". That will be discussed in the next sections, but first we consider particles and waves in physics.

1.2 Particles in Physics

Particles in physics are usually understood as special objects. To discuss the notion of particles in physics, we therefore need a theoretical understanding of the notion of "object". There is a huge literature surrounding that theme. The interested reader is referred to the fine collection of important papers concerning the philosophical and quantum physical aspects of objects in [3,2]. In her introduction Elena Castellani mentions basic philosophical questions around physical objects, the relation between part and whole, the problem of "elementary particles" as objects "without (proper) material parts", and the problem of describing the "parts of an object". Of special interest for our purpose is the problem of identity through time which ([3], p.7, Introduction by E. Castellani)

> can also be understood in the sense of searching for "something" that unites the successive parts or momentary stages of an object. In the literature, this "something" is usually called *genidentity,* after Hans Reichenbach's use of the term for characterizing the relation connecting different states of the same physical object at different times.
>
> In today's debate on physical objects, we can distinguish between two main theories of individuation, depending on whether the role of conferring individuality is ascribed to properties of the object, or to something "transcending" the object's set of properties, such as some kind of persisting substantial substratum or essence.

1.3 Waves in Physics

Waves in physics are usually described as solutions of the wave equation, in any case as functions; for example, an electro-magnetic wave is described as a function assigning to each "location" (x,y,z) and to each "point of time" t a six-tuple of values $(E_x, E_y, E_z, H_x, H_y, H_z)$ of the electro-magnetic field. If we just write down the 10 values as a 10-tuple

$$(x,\ y,\ z,\ t,\ E_x,\ E_y,\ E_z,\ H_x,\ H_y,\ H_z)$$

we do not represent, that the last six values are functionally dependent on the first four values. In the following, we do not assume any dependencies on the observed data. That allows for including non-functional waves, as for example, a water wave which is breaking near the beach; that is not representable as a function of the coordinates x,y,t - there may be more than only one surface point above (x,y) at time t. Another aspect of the "breaking water wave" is, that the uniqueness of a value at a place and a time - in that example, the z-value of the wave surface over (x,y) at time t - depends heavily on the granularity of the chosen description for "space", "time", and the domain for the "Values".

In the following, we shall employ a granular relational description of waves, clearly distinguished from functional descriptions and the description by "physical laws" (like the "wave equation"), which are well-suited for short representations of dependencies observed in granular data.

1.4 Particles, Waves, and Wave Packets

This subsection serves for describing some general similarities between particles and waves. These similarities are the key for understanding particles and waves in a semantically based granularity framework for general objects.

Abstracting from the usual physical understanding of particles to a general understanding of "objects" in "space" and "time", we now describe a property of "objects in space and time" which seems to be widely accepted, but which is not yet clear, since its granularity is not yet made explicit. We call it the

Object Axiom:
Each object is at each time granule at exactly one place.

Using the CTSOTs [19] the object axiom obviously holds, if we interpret places as states of the given CTSOT, since states are defined as the object concepts of the "actual objects". The object axiom will play the leading role for defining objects in the following.

Similarly, we describe a widely accepted property of waves by the following

Wave Axiom:
Each wave has at each place and each time granule exactly one value.

These two axioms seem to describe totally different entities, but we shall see, that they can be understood as two extremes of the same kind of entities, which will be called "wave packets" or for short "packets". For introducing them we first define Conceptual Semantic Systems in the next section.

2 Conceptual Semantic Systems

In this section we introduce Conceptual Semantic Systems (CSS) with the purpose to use semantical granularity tools symmetrically for all "dimensions" as for example for "space", "time" and also for "general objects". That end can be attained by introducing a formal representation of "instances" which are interpreted as "judgments" about an observed "reality".

For the purpose of representing scales like, for example, the scales "plant - animal - living being" or "morning - day - evening - night", or "place - town - country" we employ formal contexts, called "semantic scales". The formal concepts of these semantic scales are used as basic elements of "instances" representing observations. If these instances are taken as formal objects of a many-valued context whose values are the formal concepts of these semantic scales, then we get a "Conceptual Semantic System" as described in Definition 1.

In Definition 1 we use the notion of an ordinal context (G, M, $(W_m, \leq_m)_{m\in M}$, I) as defined in [8], where G and M are sets, $(W_m, \leq_m)_{m\in M}$ is a family of ordered sets, and I is a ternary relation with $I \subseteq \bigcup_{m\in M} G \times \{m\}\times W_m$ such that for each $g \in G$ and $m \in M$ there is exactly one $w \in W_m$ with $(g, m, w) \in I$.

Instead of arbitrary ordered sets $(W_m, \leq_m)_{m\in M}$, we choose the concept lattices of the semantic scales for defining "Conceptual Semantic Systems" in Definition 1.

Definition 1. ***"Conceptual Semantic System"***
Let M be a set and, for each $m \in M$, let $\mathbb{S}_m := (G_m, N_m, I_m)$ be a formal context. Let $(\mathbb{B}_m, \leq_m)$ be the concept lattice of $\mathbb{S}_m$. Then each ordinal context

$$\mathfrak{K} := (G, M, (\mathbb{B}_m, \leq_m)_{m \in M}, I)$$

is called a Conceptual Semantic System (CSS) with semantic scales $\mathbb{S}_m$ $(m \in M)$. The elements of G are called instances.

Remark:

- Our interpretation of the elements $m \in M$ differs remarkably from the usual "measurement meaning". We interpret the elements of M as classes, for example as classes of types for "general objects", "space", and "time". For each instance g, its family $(m(g))_{m \in M}$ is understood as "a short form of a judgment" which combines some concepts of the semantic scales.
- For ordinal contexts we have to define what we mean by the "derived context". In the following Definition 2 we define a derived context which we call the "semantically derived context".

Definition 2. ***"Semantically Derived Context"***
Let $(G, M, (\mathbb{B}_m, \leq_m)_{m \in M}, I)$ be a Conceptual Semantic System with semantic scales (G_m, N_m, I_m) $(m \in M)$ and let $int(c)$ denote the intent of a concept c. Then the formal context

$$\mathbb{K} := (G, \{(m,n) | m \in M, n \in N_m\}, J) \text{ where}$$
$$gJ(m,n) :\Longleftrightarrow n \in int(m(g))$$

is called the semantically derived context of $(G, M, (\mathbb{B}_m, \leq_m)_{m \in M}, I)$.

In the next section we use Conceptual Semantic Systems for investigating particles and waves.

3 Packets, Time, Location

In this section we give a formal definition of "particles" and "waves" in a given Conceptual Semantic System with three specified semantic scales describing *types* of "general objects" (or "packets"), "time granules", and "space granules".

3.1 Semantic Scales for Packets, Time, and Location

Definition *3.* ***"Spatio-Temporal Conceptual Semantic System"***
Let $\mathfrak{K} := (G, M, (\mathbb{B}_m, \leq_m)_{m \in M}, I)$ be a Conceptual Semantic System with semantic scales $\mathbb{S}_m = (G_m, N_m, I_m)$ $(m \in M)$ and let $P, T, L \in M$. Then the pair $(\mathfrak{K}, (P, T, L))$ is called a spatio-temporal Conceptual Semantic System.
The elements of $\mathbb{B}_P$, $\mathbb{B}_T$, $\mathbb{B}_L$ are called types of "general objects" or "packets", types of "time granules", and types of "space granules" respectively.
Let $\mathbb{K}$ denote the semantically derived context of $\mathfrak{K}$, $\underline{\mathfrak{B}}$ its concept lattice and γ its object concept mapping. For each $m \in M$ the context

$$\mathbb{K}_m := (G, \{(m,n) | n \in N_m\}, J \cap (G \times \{(m,n) | n \in N_m\}))$$

is called the m-part of $\mathbb{K}$. *Its concept lattice is denoted by* $\underline{\mathfrak{B}}_m$, *its object concept mapping by* γ_m.
The formal concepts of $\underline{\mathfrak{B}}_P$, $\underline{\mathfrak{B}}_T$, $\underline{\mathfrak{B}}_L$ *are called "general objects" or "packets", "time granules", and "space granules" of* $(\mathfrak{K}, (P, T, L))$ *respectively.*

For $m \in M$, the concept lattice $\underline{\mathfrak{B}}_m$ can be supremum-embedded into $\underline{\mathbb{B}}_m$, the concept lattice of $\mathbb{S}_m$.

Definition 4. ***"Location of a Packet at a Time Granule"***
Let $\mathfrak{C} := (\mathfrak{K}, (P, T, L))$ *be a spatio-temporal CSS. Let* $\boldsymbol{p} := (A_{\boldsymbol{p}}, B_{\boldsymbol{p}})$ *be a packet of* $\mathfrak{C}$, *and* $\boldsymbol{t} := (A_{\boldsymbol{t}}, B_{\boldsymbol{t}})$ *be a time granule of* $\mathfrak{C}$. *Then the pair* $(\boldsymbol{p}, \boldsymbol{t})$ *is called an actual packet of* $\mathfrak{C}$ *and the set*

$$L(\boldsymbol{p}, \boldsymbol{t}) := \{\gamma_L(g) | g \in A_{\boldsymbol{p}} \cap A_{\boldsymbol{t}}\}$$

is called the location of $(\boldsymbol{p}, \boldsymbol{t})$ *in* $\mathfrak{C}$ *or the location of the packet* $\boldsymbol{p}$ *at time granule* $\boldsymbol{t}$.

In biology, the habitat of a species can be understood as the location of a packet representing the species as a general object. The locations will be used to define particles and waves in the next section.

3.2 Particles and Waves as Special Packets

The following definition introduces the mathematical notions of *particles* and *waves* in spatio-temporal Conceptual Semantic Systems.

Definition 5. ***"Particles and Waves"***
Let $\mathfrak{C} := (\mathfrak{K}, (P, T, L))$ *be a spatio-temporal CSS. An actual packet* $(\boldsymbol{p}, \boldsymbol{t})$ *of* $\mathfrak{C}$ *is called*

- *not observed if* $|L(\boldsymbol{p}, \boldsymbol{t})| = 0$,
- *particle-like (or object-like) if* $|L(\boldsymbol{p}, \boldsymbol{t})| = 1$,
- *wave-like if* $|L(\boldsymbol{p}, \boldsymbol{t})| \geq 2$.

A packet $\boldsymbol{p}$ *of* $\mathfrak{C}$ *is called*

- *a particle (or an object) of* $\mathfrak{C}$ *if* $|L(\boldsymbol{p}, \boldsymbol{t})| \leq 1$ *for all time granules* $\boldsymbol{t} \in \gamma_T(G)$,
- *a wave of* $\mathfrak{C}$ *if* $|L(\boldsymbol{p}, \boldsymbol{t})| \geq 2$ *for all time granules* $\boldsymbol{t} \in \gamma_T(G)$.

The set $\gamma_T(G)$ is interpreted as the set of time granules which are "mentioned in the instances of G".

4 Representation of Classical Particles and Waves

In this section we describe first "classical particles" as particles in a spatio-temporal CSS, and then "classical waves" as waves in a spatio-temporal CSS.

4.1 Representation of Classical Particles

Let P_c, T_c, X_c be sets, interpreted as sets of "classical" particles, time points, and places, respectively. According to the "Object Axiom" (in 1.4) we introduce foreach $p \in P_c$ a mapping

$$x_p : T_c \to X_c$$

called the *trajectory of p*; $x_p(t)$ is interpreted as the place of p at time t. The structure $(P_c, T_c, X_c, (x_p|p \in P_c)\ )$ is called a *family of classical particles.*

We shall demonstrate, that each family of classical particles can be mathematically represented in a spatio-temporal CSS $\mathfrak{C}_1$ in such a way, that the classical particles are represented by particles in $\mathfrak{C}_1$. The simple idea for its construction can be seen in Table 1.

Table 1. A data table for classical particles

instances	P	T	L
(p,t)	p	t	$x_p(t)$

For the formal construction of a suitable spatio-temporal CSS $\mathfrak{C}_1$:= ((G, M, $(\mathbb{B}_m, \leq_m)_{m \in M}$, I), (P,T,L)) we introduce the set $G := P_c \times T_c$, define P, T, L as mappings from G into P_c, T_c, and X_c, respectively, by
P(p,t) := p, T(p,t) := t, L(p,t) := $x_p(t)$. Let M := $\{P,T,L\}$; the semantic scales are chosen as the nominal scales

$\mathbb{S}_P := (P_c, P_c, =)$, $\mathbb{S}_T := (T_c, T_c, =)$, $\mathbb{S}_L := (X_c, X_c, =)$.

The ternary incidence relation I is defined by

I := $\{((p,t), m, \gamma_m(m(p,t))|p \in P_c, t \in T_c, m \in M)\}$.

The following Lemma 1 shows, that the classical particles are represented as particles in $\mathfrak{C}_1$.

Lemma 1. ***Classical Particle Representation***
*Let $(P_c, T_c, X_c, (x_p|p \in P_c)\ )$ be a family of classical particles and $\mathfrak{C}_1$ its corresponding spatio-temporal CSS as defined above. Then for each classical particle $p \in P_c$ the packet **p**:= $\mu_P(P,p)$ is a particle of $\mathfrak{C}_1$, where $\mu_P(P,p)$ denotes the attribute concept of (P,p) in $\mathbb{K}_P$.*

Proof. It is obvious, that $\mathfrak{C}_1$ is a spatio-temporal CSS. Let $p \in P_c$. To prove, that the packet **p**:= $\mu_P(P,p)$ is a particle of $\mathfrak{C}_1$, let $\mathbf{t} \in \gamma_T(G)$. Then there exist $(q,t) \in G = P_c \times T_c$ such that $\mathbf{t} = \gamma_T(q,t)$. Hence $\gamma_T(q,t) = \mu_T(T,t) = \gamma_T(p,t)$ and L(**p**,**t**)= $\{\gamma_L(g)|g \in A_\mathbf{p} \cap A_\mathbf{t}\} = \{\gamma_L(p,t)\}$. Hence **p** is a particle in $\mathfrak{C}_1$. □

4.2 Representation of Classical Waves

Similar to the representation of a family of classical particles we can also mathematically represent classical waves. The main idea is very simple: Let's assume, that we wish to describe a "usual" wave, namely a function f defined on the real plane, which assigns to each point (x,y) and to each time point t an amplitude value z := f((x,y),t). For each time point t this function can be described by the "family" of contour lines of all levels of f. This is the main similarity between a family of classical particles and a wave as a family of contour lines. That leads to a representation of a wave as a packet, which is a superconcept of all object concepts in a nominal semantic scale for the levels of f. That is explained in the following Lemma 2, which shows, that according to the Wave Axiom each wave, which is described as a function, can be represented in a spatio-temporal CSS $\mathfrak{C}_2$ as a wave of $\mathfrak{C}_2$.

Lemma 2. Classical Wave Representation
Let a "classical wave" be described as a function

$$f : X_c \times T_c \to Z_c$$

whereX_c*,* T_c*,* Z_c *are sets, interpreted as sets of "classical" places, time points, and values, respectively.*
For the formal construction of a suitable spatio-temporal CSS $\mathfrak{C}_2 := (G, M, (\mathbb{B}_m, \leq_m)_{m\in M}, I), (P, T, L))$ *we introduce the set* $G := X_c \times T_c$*, define* P*,* T*,* L *as mappings from* G *into* Z_c*,* T_c*, and* X_c *respectively, by* $P(x,t) := f(x,t), T(x,t) := t, L(x,t) := x$*. Let* $M := \{P, T, L\}$*; the semantic scales are chosen as the nominal scales* $\mathbb{S}_P := (Z_c, Z_c, =)$*,* $\mathbb{S}_T := (T_c, T_c, =)$*,* $\mathbb{S}_L := (X_c, X_c, =)$ *and* $I := \{((x,t), m, \gamma_m(m(x,t))|(x,t) \in G, m \in M)\}$*. Then* $\mathfrak{C}_2$ *is a spatio-temporal CSS. Let* $\mathbf{p}$ *denote the top element in the concept lattice* $\mathfrak{B}_P$ *of the P-part of the semantically derived context* $\mathbb{K}$*.*
Then $\mathbf{p}$ *is a wave in* $\mathfrak{C}_2$*, if* $|X_c| \geq 2$*. Otherwise,* $\mathbf{p}$ *is a particle.*

Proof. It is obvious, that $\mathfrak{C}_2$ is a spatio-temporal CSS. Since $\mathbf{p}$ is the top element of $\mathfrak{B}_P$ its extent $A_\mathbf{p}$ = G. To prove, that the packet $\mathbf{p}$ is a wave of $\mathfrak{C}_2$, let $\mathbf{t} = (A_\mathbf{t}, B_\mathbf{t}) \in \gamma_T(G)$, say $\mathbf{t} = \gamma_T(x,t)$ for some $x \in X_c$ and $t \in T_c$. Hence the extent $A_\mathbf{t} = X_c \times \{t\}$. Then L(**p**,**t**)= $\{\gamma_L(g)|g \in A_\mathbf{p} \cap A_\mathbf{t}\} = \gamma_L(A_\mathbf{t})$, since $A_\mathbf{p} = G$. Hence L(**p**,**t**)= $\gamma_L(X_c \times \{t\})$. Therefore, $|L(\mathbf{p},\mathbf{t})| \geq 2 \iff |X_c| \geq 2$. □

In the next section we present three examples, two for classical particles and one for a packet, which sometimes behaves like a wave, and sometimes behaves like a particle. That will be understood as the usual behavior of packets.

5 Examples: Harmonic Oscillator, Practicing Gymnasts, Hare and Hedgehog

The first example represents a simple harmonic oscillator on the unit circle as a particle, the second one represents a well-known gymnastic demonstration for three gymnasts as a family of particles, where their exercises also behave like particles. The third example represents in the German tale "Der Wettlauf zwischen dem Hasen und dem Igel" (The Race Between the Hare and the Hedgehog) the "species" of hedgehogs as a packet which is neither a particle nor a wave.

For some of these examples we employ in their graphical representations arrows indicating transitions based on the usual linear order on the time set. For the general introduction of transitions in CTSOTs the reader is referred to [18,22]. Transitions for spatio-temporal CSSs in general will be investigated in future research.

5.1 A Harmonic Oscillator

We construct a simple harmonic oscillator as a family with a single classical particle $\mathfrak{H} := (P_c, T_c, X_c, (x_p|p \in P_c))$, where $P_c := \{p\}$ has only one element p, $T_c := [0, 2\pi]$, $X_c := \{(x,y) \in \mathbb{R}^2 | x^2 + y^2 = 1\}$, and for $t \in T_c$ let $\mathbf{x}_p(t) := (\cos(t), \sin(t))$. The corresponding spatio-temporal CSS $\mathfrak{C}_1(\mathfrak{H})$ clearly has an infinite semantically derived context. For visualizing its concept lattice we restrict the set T_c to the set $T_{12} := \{\frac{2\pi k}{12} | k \in \{0, 1, ..., 12\}\}$. The concept lattice of the semantically derived context of the restricted system is shown in

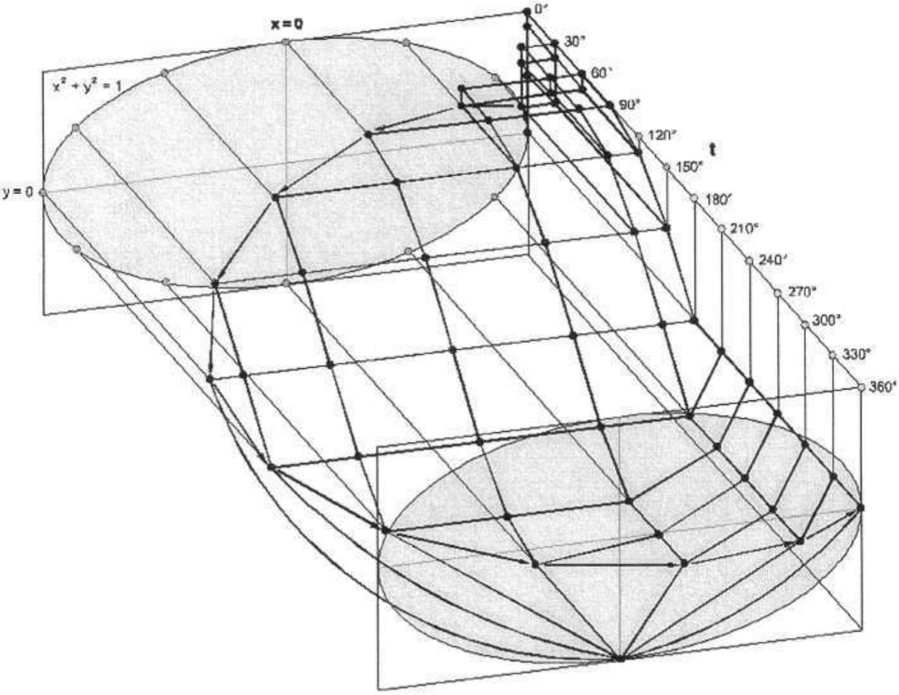


Fig. 1. A harmonic oscillator as a particle moving on the unit circle (drawn by Julia Wiskandt)

Figure 1, which represents the movement of a particle on the unit circle in the situation space, that is the direct product of the state space (here the unit circle) and the time (here the set T_{12}).

5.2 Three Gymnasts: Roll and Jump

We now construct a spatio-temporal CSS, which represents a well-known gymnastic demonstration for three gymnasts, say Tobias, Konstantin, and Florian. They start their demonstration at time point 0 standing on a line, Tobias on place1, Konstantin on place2, and Florian on place3. The main information for them is, that the gymnast on the middle place2 rolls to an outer place, while at the same time the gymnast on that outer place whereto the "inner" gymnast rolls, jumps straddling over the rolling gymnast. Each gymnast, which has reached an outer place, has to turn around for the next jump. Konstantin starts rolling to place1 and Tobias jumps straddling to place2. A whole period of this demonstration is shown in Figure 2, where the life track of each gymnast is represented by a sequence of arrows, bold for Tobias, middle for Konstantin, and thin for Florian. The formal objects from 1 to 21 are the "instances" of the underlying spatio-temporal CSS. The reader will be convinced, that the three gymnasts will be represented as three particles of the underlying spatio-temporal CSS. But in this example, there are more formal particles, namely the exercises "roll", "jump", and "turn around" yield, as values of the many-valued attribute "exercise", also particles, if we choose as the "P-part" not the gymnasts, but the exercises. Instead of explaining the details, we just show the corresponding transition diagram for the exercises in Figure 3.

For example, instance 9 says, that Konstantin at placel and time point 1 has turned around; if we now follow the "turn around arrow" to the object concept of the instance 3, we see from Figure 2 that someone has turned around at place3 and time point 2, and that is Tobias, since he is the only gymnast, who is at time point 2 on place3.

The observation, that the exercises "behave" sometimes as particles, as in this example, and sometimes as waves, led the author to the present paper. In the next example we will find a typical "wave packet" which sometimes behaves like a wave and sometimes like a particle.

5.3 Hare and Hedgehog

The following example has been introduced by the author [21].

> The German tale "Der Wettlauf zwischen dem Hasen und dem Igel" (The Race Between the Hare and the Hedgehog) [1] contains a nice and well-known misinterpretation on the side of the hare, which is based on the fact that the hare does not distinguish between the hedgehog and its wife.

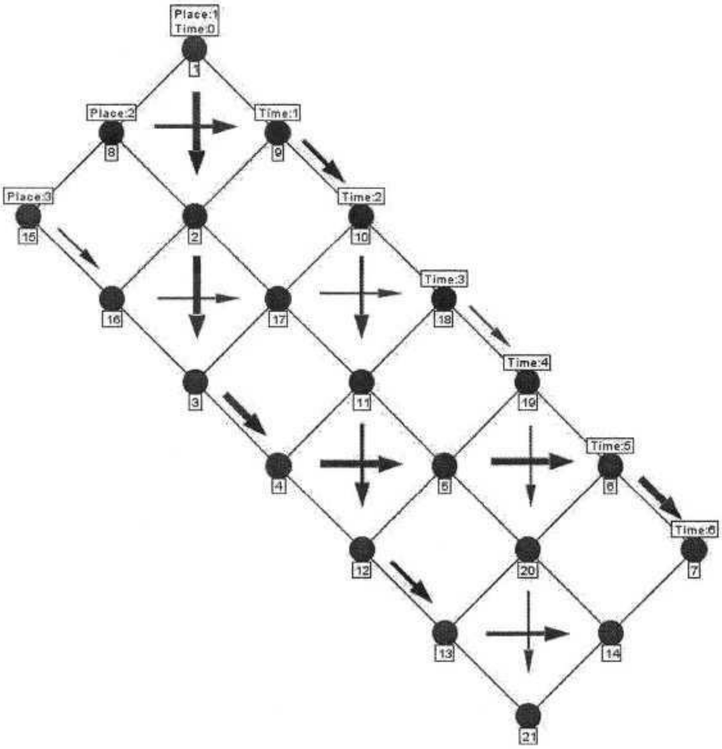


Fig. 2. Tobias (bold arrow), Konstantin (middle), Florian (thin)

A Short Version of the Tale

On a nice Sunday morning the hedgehog leaves his family for a walk to his field where he meets the hare. They decide to run a race. The hedgehog asks his wife at home to help him in that race by waiting at the lower end of the field. Then the hedgehog starts the race together with the hare at the upper end of the field, but after a few steps the hedgehog stops and returns to the starting point. The hare does not know that and runs down to the lower end of the field where the wife of the hedgehog is waiting. Since the hare does not distinguish her from the hedgehog, he thinks that the hedgehog was quicker. He repeats the race until he breaks down and dies.

Now we demonstrate a simple example of a "wave packet". We make it explicit as a packet in a spatio-temporal CSS constructed from the first part of the race between the hare and the hedgehog as represented in Table 3. For that purpose we introduce semantic scales for P := animal, T := time, L := place,

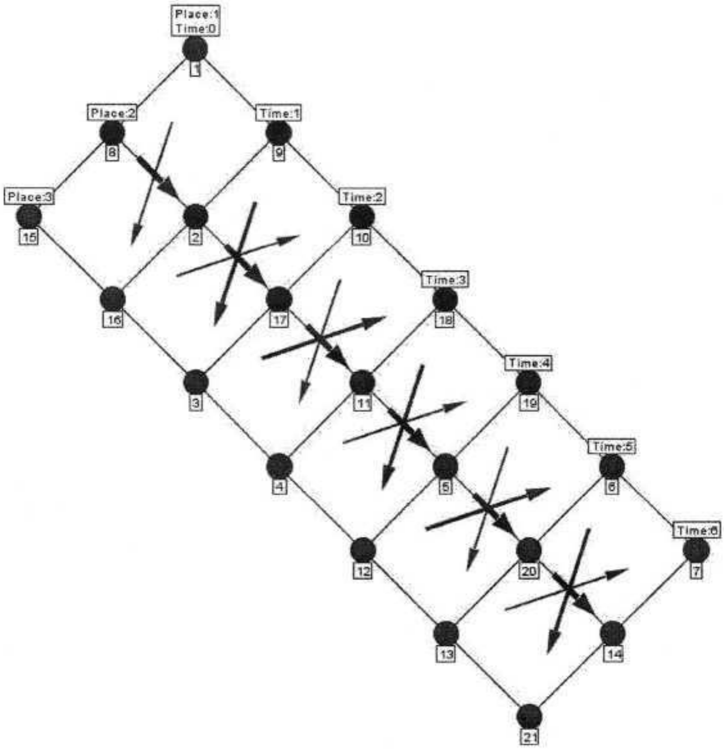


Fig. 3. Exercises as particles: roll(bold arrows), turn around (middle), jump (thin)

taking $\mathbb{S}_P$ as the nominal scale for the set $\{hare, hedgehog\}$, $\mathbb{S}_T$ as the nominal scale for the set $\{0, 1, 2, 3\}$, and the semantic scale $\mathbb{S}_L$ as given in Table 2.

These semantic scales together with the many-valued context shown in Table 3 form the spatio-temporal CSS. For example, in Table 3 the value “hedgehog” denotes the attribute concept of the attribute “hedgehog” in $\mathbb{S}_P$. We interpret instance 1 (and instance 2) in Table 3 as the statement “An animal of type hedgehog was observed at time 0 in the House.” As in many applications multiple observations are protocolled here. That appears also, if we restrict a many-valued context to some part of it, as we did here, since we do not use here the attribute “gender” of the original many-valued context in [21].

The “hare packet” $\mathbf{p_1} := \mu_P(animal, \boldsymbol{hare})$, that is the attribute concept of (animal,hare) in the P-part of the semantically derived context $\mathbb{K}_P$ has at time granule $\mathbf{0} := \mu_T(time, 0)$ the location $\mathbf{L}(\mathbf{p_1}, \mathbf{0}) = \emptyset$; that means, that the hare was not mentioned at time granule $\mathbf{0}$. But for the other three time granules the hare packet behaves particle-like, hence the hare packet $\mathbf{p_1}$ is a particle in this spatio-temporal CSS. Using a time relation as in CTSOTs, we could introduce transitions and life tracks of objects in spatio-temporal CSSs. That will be done in a more general setting in a future paper.

Table 2. The semantic scale $\mathbb{S}_L$ for the places

	House	Field	Upper Field	Lower Field
House	×			
Field		×		
Upper Field		×	×	
Lower Field		×		×

The "hedgehog packet" $\mathbf{p_2} := \mu_P(animal, hedgehog)$, that is the attribute concept of (animal,hedgehog) in the P-part of the semantically derived context $\mathbb{K}_P$ has at time granule $\mathbf{0}:= \mu_T(time, 0)$ the location L($\mathbf{p_2}$,$\mathbf{0}$) $= \{\gamma_L(1)\}$, hence the actual packet ($\mathbf{p_2}$,$\mathbf{0}$) is particle-like, while ($\mathbf{p_2}$,$\mathbf{1}$) is wave-like. That can be seen easily in Figure 4, where the "oscillating" location of the "hedgehog packet" at the four time granules is indicated by black squares at the object concepts of the instances 1,2,4,5,7,8,10,11 where "hedgehogs are mentioned". That yields a simple conceptual understanding of the phenomena of "generation of particles from waves" and the "decay of particles into waves". Hence, the "hedgehog packet" is a "proper wave packet", that is a packet, which is neither a particle nor a wave.

The example of the "hedgehog packet" can be easily extended to many other examples in daily life. That may help "de-mystifying" and understanding better the "wave packets" in physics.

Table 3. A many-valued context describing the first part of the race

	P	T	L
instance	animal	time	place
1	hedgehog	0	House
2	hedgehog	0	House
3	hare	1	Field
4	hedgehog	1	Field
5	hedgehog	1	House
6	hare	2	Field
7	hedgehog	2	House
8	hedgehog	2	House
9	hare	3	Upper Field
10	hedgehog	3	Upper Field
11	hedgehog	3	Lower Field

6 Conclusion and Future Research

We have introduced a "symmetric" mathematical representation of objects, space, and time in continuously fine or arbitrarily coarse granularity. The intu-

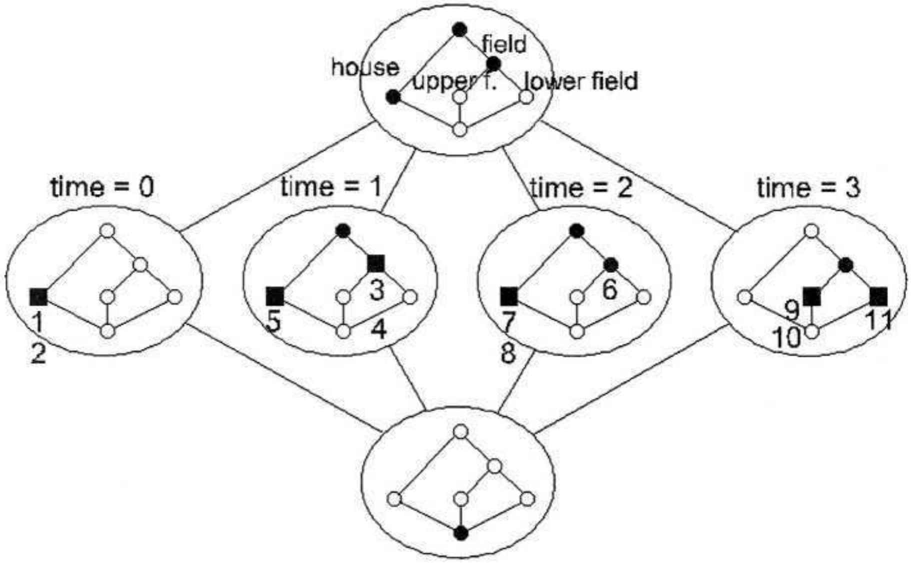


Fig. 4. The "species" hedgehog as a "wave packet" indicated by squares

itive notion of "wave packets" has been conceptually described by the definition of *general objects* or *packets* in *spatio-temporal Conceptual Semantic Systems.* That led to a mathematical definition of*particles* and *waves,* which describes the "classical" particles and waves in physics. In the examples we have shown, how the developed tools can be used for describing discrete phenomena of movements of general objects.

Future research might focus on the following three topics.

1. The development of a spatio-temporal theory of *general objects* including a generalization of the notions of transitions and life tracks as they have been introduced by the author for the special case of Conceptual Time Systems with Actual Objects and a Time Relation (CTSOT).
2. The investigation of the connection of Conceptual Semantic Systems to Contextual Logic as developed by R. Wille [13].
3. The development of a spatio-temporal logic combining Contextual Logic with the Temporal Logic as developed by D. Gabbay [6] and J. van Benthem [10].

Acknowledgement. I would like to express my gratitude to Walter Sedelow for his encouragement and help during the development of Temporal Concept Analysis; I also thank Rudolf Wille for his philosophical and mathematical support in numerous discussions; and last but not least I would like to thank my dear wife Margit for her patience and solicitude.

References

1. Bechstein, L.: Der Wettlauf zwischen dem Hasen und dem Igel. In: Ludwig Bechsteins Märchenbuch. F.W. Hendel Verlag Leipzig 1926, 260-264.
2. Butterfield, J. (ed.): The Arguments of Time , Oxford University Press, 1999.
3. Castellani, E.(ed.): Interpreting Bodies: Classical and Quantum Objects in Modern Physics. Princeton University Press 1998.
4. Einstein, A.: Zur Elektrodynamik bewegter Körper. Annalen der Physik 17 (1905): 891-921.
5. Ganter, B., R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin-Heidelberg-New York 1999.
6. Gabbay, D.M., I. Hodkinson, M. Reynolds: Temporal Logic - Mathematical Foundations and Computational Aspects. Vol.1, Clarendon Press Oxford 1994.
7. Reichenbach, H.: The Direction of Time. Edited by M. Reichenbach. Berkeley: University of California Press, 1991. (Originally published in 1956.)
8. Strahringer, S., Wille, R.: Towards a Structure Theory for Ordinal Data. In: M. Schader (ed.): Analysing and Modelling Data and Knowledge. Springer, 1992; 129-139.
9. Toraldo di Francia, G.: A World of Individual Objects? In: Castellani, E.(ed.): Interpreting Bodies: Classical and Quantum Objects in Modern Physics. Princeton University Press 1998, 21-29.
10. van Benthem, J.: Temporal Logic. In: Gabbay, D.M., C.J. Hogger, J.A. Robinson: Handbook of Logic in Artificial Intelligence and Logic Programming. Vol. 4, Epistemic and Temporal Reasoning. Clarendon Press, Oxford, 1995, 241-350.
11. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.): Ordered Sets. Reidel, Dordrecht-Boston 1982, 445-470.
12. Wille, R.: Introduction to Formal Concept Analysis. In: G. Negrini (ed.): Modelli e modellizzazione. Models and modelling. Consiglio Nazionale delle Ricerche, Instituto di Studi sulli Ricerca e Documentatione Scientifica, Roma 1997, 39-51.
13. Wille, R.: Contextual Logic summary. In: G. Stumme (ed.): Working with Conceptual Structures. Contributions to ICCS 2000. Shaker, Aachen 2000, 265-276.
14. Wolff, K.E.: A first course in Formal Concept Analysis - How to understand line diagrams. In: Faulbaum, F. (ed.): SoftStat '93, Advances in Statistical Software 4, Gustav Fischer Verlag, Stuttgart 1994, 429-438.
15. Wolff, K.E.: Concepts, States, and Systems. In: Dubois, D.M. (ed.): Computing Anticipatory Systems. CASYS'99 - Third International Conference, Liège, Belgium, 1999, American Institute of Physics, Conference Proceedings 517, 2000, pp. 83-97.
16. Wolff, K.E.: Towards a Conceptual System Theory. In: B. Sanchez, N. Nada, A. Rashid, T. Arndt, M. Sanchez (eds.): Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, SCI 2000, Vol. II: Information Systems Development, International Institute of Informatics and Systemics, 2000, 124-132.
17. Wolff, K.E.: Temporal Concept Analysis. In: E. Mephu Nguifo et al. (eds.): ICCS-2001 International Workshop on Concept Lattices-Based Theory, Methods and Tools for Knowledge Discovery in Databases, Stanford University, Palo Alto (CA), 91-107.
18. Wolff, K.E.: Transitions in Conceptual Time Systems. In: D.M.Dubois (ed.): International Journal of Computing Anticipatory Systems vol. 11, CHAOS 2002, 398-412.

19. Wolff, K.E.: Interpretation of Automata in Temporal Concept Analysis. In: U. Priss, D. Corbett, G. Angelova (eds.): Integration and Interfaces. Tenth International Conference on Conceptual Structures. Lecture Notes in Artificial Intelligence, 2393, Springer-Verlag (2002), 341-353.
20. Wolff, K.E., W. Yameogo: Time Dimension, Objects, and Life Tracks - A Conceptual Analysis. In: A. de Moor, W. Lex, B. Ganter (eds.): Conceptual Structures for Knowledge Creation and Communication. LNAI 2746, Springer. Heidelberg 2003, 188-200.
21. Wolff, K.E.: Towards a Conceptual Theory of Indistinguishable Objects. In: Eklund, P. (ed.): Concept Lattices: Proceedings of the Second International Conference on Formal Concept Analysis. LNCS 2961, Springer-Verlag, 2004, 180-188.
22. Wolff, K.E.: Temporal Concept Analysis - States, Transitions, Life Tracks. Preprint Darmstadt University of Applied Sciences, Mathematics and Science Faculty, 2004.

A Void Type for Medical Language Disambiguation

Christian Jacquelinet

Département Médical et Scientifique,
Etablissement français des Greffes, 5, rue Lacuée,
75012 Paris, France
christian.jacquelinet@univ-rennes1.fr

Abstract. This paper focuses on lexical semantics, polysemy and disambiguation in the perspective of medical language understanding. Previous results obtained in the domain of transplantation and organ failure with the semantic analysis of medical terms prompt us to check the capabilities of our prototype and its underlying semiotic model to deal with ambiguities and polysemy. We show how experimentations around linguistic observations lead us to introduce a void type to respect lexical ambiguities when the co-text is not sufficient for disambiguation. Then we demonstrate how the "creative use of words" (i.e. new senses in novel context) imposes to dynamically allocate types and roles according the co-text, using a type contraction operation.

1 Introduction

Despite many works performed about Natural Language Processing of medical text reports [1-4], knowledge acquisition from medical literature [5-6], automated indexing [7-9] and linguistic knowledge acquisition [10], Medical Language Understanding (MLU) remains a difficult and disappointing research domain: after fifty years, results obtained remain very limited when compared to human genomics [11]. Nevertheless, efforts must continue to improve theories and to provide more efficient tools for MLU. The fact is that health care delivery is dependent on accurate and detailed clinical data that is mostly contained within medical text reports [12], but clinical data records do not support the richness of free text medical reports. Recording clinical data in a database produces a semantic alteration that has been shown to affect the evaluation of medical practices [13].

Taking the lessons from the past, some misleading approaches can be avoided. The first misleading principle is to process sequentially to some *text-zoning,* splitting the text into a set of text segments; then to some *pre-processing* turning text segments into lexical entries and finally to *syntactic parsing* transforming sequences of lexical entries into a set of parsed trees. *Semantic analysis* that generates semantic or logical structures from parsed trees, co-reference resolution and discourse analysis which attempt to identify the different descriptions of the same entities in different parts of the text constitute the ultimate treatments.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 142–155, 2004.

This classical sequential approach [14] is misleading for two main reasons: (i) syntactical analysis frequently fails to handle ambiguities without semantic or pragmatic knowledge and (ii) semantic analysis is frequently unable to choose between concurrent parsed trees without lexical knowledge. Furthermore, two sentences with distinct meanings and distinct semantic structures can share same syntactic structure: this was pointed out by L. Wittgenstein in [15]. A second misleading principle, also a consequence of the sequential approach, is the use of multiple knowledge representational formalisms (KRF) for each component. LU requires morphological, lexical, syntactical, semantic and pragmatic meta-knowledge that is generally stored in various Knowledge Bases, lexicons, grammars or ontologies. A third misleading understatement is due to the use of sense enumerative lexicons, unable to deal with “the creative use of words” (words have new senses in novel contexts), the “permeability of word senses” (word senses are not atomic definitions but overlap to other senses) and the “expression of multiple syntactic forms” (a single word sense can have multiple syntactic realization) as shown by J. Pustejovsky [16].

Previous results obtained with the semantic analysis of medical terms in the domain of transplantation and organ failure [17] prompt us to check the capabilities of our prototype, named RIBOSOME, to deal with ambiguities and polysemy. Starting from linguistic observation, we attempt to demonstrate how the respect of ambiguities when the co-text is not sufficient for disambiguation implies to introduce a void type in the concept type lattice. Then we show how the “creative use of words” (i.e. new senses in novel context) imposes to dynamically allocate type, categories and roles with the co-text.

2 Material and Methods

2.1 A Semiotic Knowledge Representational Model for MLU

A first cognitive requirement for our model is that any kind of knowledge share the same KRF. Starting from the CG formalism for knowledge representation, we then considered semiotics and lexical semantic principles to design a final extended model integrating linguistic and domain knowledge in a unified representational formalism reported in [18]. A second cognitive and linguistic requirement is that syntactic and semantic analysis of texts constitute the same cognitive process: not only the categorization of *organ* as a noun or as a musical instrument can rely on a unique lexical recognition analytic process, but also the compositionality of syntactic elements and the compositionality of atoms of meaning into more complex syntactical or semantic molecules can also rely on a unique cognitive synthetic processing. Our KRF adds to the Conceptual Graph (CG) model [19] a major extension: concepts and conceptual relations (CR) are associated to *lexia* that can be words, part of words, locutions or any kind of sign, even the blank space between words. The basic structure of the Knowledge Bases (KB) is referred to as a *cognitive sign* (CS).

A CS associates a type label **t**, a set of lexia *L,* a contextual marker *#i,* a couple of valences (v_λ, v_ρ) that specifies the maximum number of refined and refining cognitive signs that can be linked during the semantic analysis and a couple of positional indicators (p_λ, p_ρ) that specifies the relative position of the lexia of refined and refining cognitive signs:

$$\text{CS} = [\mathbf{t}\text{: } L\ (v_\lambda, v_\rho)\ (p_\lambda\ , p_\rho)\text{: \#i}].$$

A given type label **t** can appear in many syntagmatic contexts identified by their contextual marker #i. A valence v is an integer or is noted n when there is no limitations to the number of possible conceptual refinings. A (1, 1) cognitive sign accepts only one refined cognitive sign and one refining cognitive sign during text processing. A (1, n) cognitive sign refines only one cognitive sign and accepts n refining cognitive signs. Positional indicators are noted =, > or < weather *lexia* of refining or refined cognitive signs are respectively anywhere, after or before *lexia* of the given cognitive sign. The most common value of positional indicators (p_λ, p_ρ) is (=, =) for conceptual cognitive signs, (>, >) for relational cognitive signs in French and more frequently (<, <) in English. A given *lexia* can appear in different contexts, with the same type label or with different type labels, allowing the treatment of generalized polysemy and the contextuality of meaning.

2.2 Lexical Semantics Principles

A lexia such as the noun *organ* can be recognized and labeled with:

[**Word**: {*organ*} (1, 1) (=, =): #1]
[**Noun**: {*organ*} (1, n) (=, =) : #2]
[**Musical Organ**: {*organ*} (1, n) (=, =): #3]
[**Anatomical Organ**: {*organ*} (1, n) (=, =): #4]

This example illustrates the semic decomposition of the lexia *organ* into atoms of meaning such as **Anatomical Organ** and **Musical Organ** which all are linguistic denotations of *organ* or into morpho-syntactical atoms such as **Word** or **Noun.** If required, a same *lexia* can appear in two distinct contexts sharing the same type label. For example, it is possible to enumerate a second context for *organ* categorized as an **Anatomical Organ** if a distinction between a native and a retrieved organ is required: [**Anatomical Organ:** {*organ*} (1, n) (=, =): #51]. A given type label may be shared by many lexia, allowing the re-use of knowledge models. Every word can be labeled as **Word** and stored within a unique context. A given lexia like *surgeon* can be labeled as **Noun, Person** or **Surgery**:

[**Word**: {*..., the, surgeon, he, thinks, to, a, calculus, ...*} (1, 1) (=, =): #23]
[**Noun**: {*..., surgeon, calculus, nephropathy, ...*} (1, n) (=, =): #24]
[**Verb**: {*..., thinks, ...*} (1, n) (=, =): #25]
[**Surgery**: {*..., surgeon, surgery, surgical...*} (1, n) (=, =): #26]
[**Person**: {*..., surgeon,...* } (1, n) (=, =): #27]

[**Thought**: {..., *thinks, think,* ...} (1, n) (=, =): #28]
[**Calculation**: {..., *calculus,* ...} (1, n) (=, =): #29]

In the above example, the type label **Surgery** is not a reserved denotation of the sole lexia *surgery.* It is also shared with other medical terms (surgeon, surgical) that have this meaning as a connotation. This feature permits the reuse of syntagmatic links with other cognitive signs: compositional meta-knowledge acquired around the lexia *calculus* with the cognitive label of **Noun** in the context #24 will also be valid for *surgeon.*

2.3 Compositionality Principles

A key feature in our model is the specification of conceptual relations signatures at the semiotic level: lexicalized relational cognitive signs are associated to head and tail lexicalized conceptual cognitive signs. Furthermore, valences and positional markers are also used to restrict very precisely the number of join possibilities for graphs. Positional indicators are used during semantic analysis to deal with the semantic impact of words order in messages such as *gastric tumor wall / gastric wall tumor.* Let be 2 conceptual CS #i and #k registered within the KB as a valid signature of a relational CS #j:

[$\mathbf{C}_a$: L_a ($v_{\lambda a}$, $v_{\rho a}$) (=, =): #i]-
[(RC): L_b ($v_{\lambda b}$, $v_{\rho b}$) (p_λ , p_ρ): #j]→[$\mathbf{C}_c$: L_c ($v_{\lambda c}$,$v_{\rho c}$) (=, =): #k]

if ω_a, ω_b, ω_c are respectively lexia in L_a, L_b, L_c, the semiotic graph SG$_1$:

[$\mathbf{C}_a$: ω_a: #i]→[(RC): ω_b: #j]→[$\mathbf{C}_c$: ω_c: #k]

will result from the semantic analysis (SemAn) of the lexia sequences:

$\omega_a\ \omega_b\ \omega_c$ if $(p_\lambda\ , p_\rho) \in \{(>,>), (=,=), (=,>), (>,=)\}$
$\omega_c\ \omega_b\ \omega_a$ if $(p_\lambda\ , p_\rho) \in \{(<,<), (=,=), (=,<), (<,=)\}$
$\omega_a\ \omega_c\ \omega_b$ if $(p_\lambda\ , p_\rho) \in \{(>,<), (=,=), (=,<), (>,=)\}$
$\omega_b\ \omega_a\ \omega_c$ if $(p_\lambda\ , p_\rho) \in \{(<,>), (=,=), (=,>), (<,=)\}$

The conceptual graph corresponding to SG$_1$ is obviously CG_1:
[$\mathbf{C}_a$]→(RC)→[$\mathbf{C}_c$]

2.4 Relating Lexia to Cognitive Graphs

A special auto-combining marker $ is used in relational CS to restrict the composition of head and tail CSs only to those sharing a same lexia.

Let be three CS defining a valid signature in the KB:

[$\mathbf{C}_a$: L_a ($v_{\lambda a}$, $v_{\rho a}$) (=, =): #i]-
[(CR): \$ (λ_b, ρ_b) (=, =): #j]→[$\mathbf{C}_c$: L_c (λ_c, ρ_c) (=, =): #k]

if ω is a common lexia of L_a and L_c then **SemAn**(ω) produces SG_2:
[$\mathbf{C}_a$: ω: #i]→[(CR): \$: #j]→[$\mathbf{C}_c$: ω: #k]

For example, if the KB contains:

[**Person**: {*surgeon, patient, dentist*} (1,n): #30]-
[(EXPR): $ (1,1) (=, =): #31]→[**Knowledge**: {*surgeon, dentist*} (1,n) : #32]

then, **SemAn***(surgeon)* produces:

[**Person**: *surgeon*: #30]→[(EXPR): $: #31]→[**Knowledge**: *surgeon*: #32]

This mean can be used to relate a lexia to a complex cognitive structure.

2.5 Semantic Analysis with RIBOSOME

RIBOSOME comprises a KB and a parser that turns sentences and medical texts into CGs. The KB integrates the concept type lattice, the conceptual relations hierarchy, the conceptual relations signatures (i.e. valid head and tail concepts for a given conceptual relation) and a semantic lexicon relating lexical entries to their relevant conceptual structures. The resulting KB defines a semantic network. Because this semantic network alternates lexicalized conceptual and relational cognitive signs, we referred it as to a *semiotic network.*
During the parsing of a text, the input is firstly segmented according to recognized lexia. Words composing locutions and locutions are analyzed at the same time. The second phase is a *Lexical Analysis* (LexAn) turning recognized lexia of the input into relevant cognitive signs according to lexical knowledge previously registered within the KB. LexAn consists in a contextual selection of conceptual structures, triggered by the lexical input. The production of relational cognitive signs is restricted to relational cognitive signs relating lexia existing in the co-text. The third phase, referred to as a *Knowledge Synthesis* (KnΣ), realizes the composition of cognitive signs into tripartite semiotic graphs with one relation linking two concepts. Free remaining cognitive signs are then eliminated from the analysis. Tripartite cognitive graphs are finally joined on common edges to form one or more complex semiotic graphs. Valences, positional markers and signature of relational cognitive signs registered within the KB rule the composition of cognitive signs. The semantic analysis (SemAn) of a textual input in our approach combines sequentially LexAn then KnΣ.

2.6 Linguistic Observations and Semiotic Statements

Lets consider the following short sentences:

(1) *Le chirurgien opère un calcul rénal. (⋆ the surgeon operates a renal calculus)*
(2) *Le chirurgien opère un calcul arithmétique.*
(3) *The surgeon operates on a renal calculus.*
(4) *The surgeon performs arithmetic calculation.*
(5) *He thinks to a complicated calculus.*
(6) *The teacher thinks to a renal calculus.*
(7) *The surgeon learned integral calculus.*

Calcul in French is a polysemic and ambiguous term that denotes according to the context a **lithiasic stone** (1), a **mathematic calculus** or a **calculation** (2).

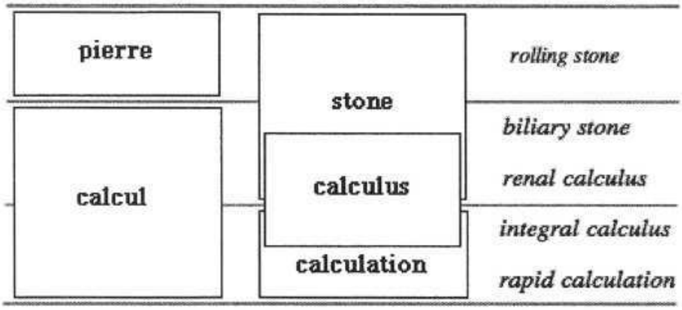


Fig. 1. Mapping *calcul* to *calculus*

Because there is a diachronic relationship between the *lithiasic calculus* and the *mathematic calculus:* calculations where performed with small stones: calculi, *calcul* is indeed a polysemic term and not an homonymic one. Its Latin root *calculus* is also used in English with the denotation of **lithiasic stone** (3) and **mathematic calculus** (4) according to the co-text. Wordnet dictionary gives three definitions: [*a*] the branch of mathematics that is concerned with limits and with the differentiation and integration of functions; [*b*] a hard lump produced by the concretion of mineral salts found in hollow organs or ducts of the body; and [*c*] an incrustation that forms on the teeth and gums.

There is no exact mapping between French and English [Figure 1]. *Calculus* in English is not frequently used to refer to biliary stones whereas the noun *pierre* in French is not used for lithiasic concretions. According to the co-text, *calcul* in French maps to *stone, calculus* or to *calculation;* and probably the simple and non contextual translations of (1) sounds odd to native english speakers. The verb operate is also polysemic, with no exact mapping from French to English. One can *operate a device* in English, another verb such as *actionner* is required in such a context in French. The post-fix *on* gives per se a surgical denotation to *operate* in (3) so that the subject and the direct object are not required to specify the type of operation. Indeed with (5), (6) and (7), it is not because the agent of the action is a *surgeon* that the *calculus* is a **stone**. As far as somebody can do something related to a calculus without further precision (5), the semantic representation of *calculus* should preserve the ambiguity.

3 Results

3.1 Introducing a Void Type for Progressive Disambiguation

To preserve ambiguities without generating artificial forests of semantic trees, we introduce a void type noted ∅ used to connect unresolved ambiguities to their refining concepts during the semantic analysis. In our case, *calculus* and *calcul* would be 2 valid lexia for ∅:

[∅ : {..., *calculus, calcul,...*} (n, n): #5]

If *complicated* adds to *calculus* refining syntagmatic information but does not clarify the type of calculus, then the refining information will be preserved if one stores in the semiotic network:

[∅ : {*..., calculus, calcul,...*} (n, n): #5]-
  [(REL): _ (<,<) (1, 1) : #6]→[**Complication**: {*.., complicated,...*} (1, n): #8]
  [(REL): _ (>,>) (1, 1) : #7]→[**Complication**: {*.., compliqué,...*} (1, n): #9]

In this simple cognitive grammar, we use a generic CR (REL) and *complicated* is considered as monosemic. This example also pinpoints the influence of positional markers on conceptual composition. The French adjective *compliqué* is composed with *calcul* as a refining concept when the adjective is featuring after the noun and when there is a blank space (noted: _) in between. Similarly, the English adjective *complicated* is composed with *calculus* as a refining concept when the adjective is featuring before the noun and when there is a blank space in between.
If one considers that *rénal* or *arithmétique* disambiguates the type of *calcul,* then we can introduce a CR (TYPE) to specify the type. Accordingly, the semiotic network can be completed as follow:

[∅ : {*..., calculus, calcul,...*} (n, n): #5]-
  [(REL): _ (<,<) (1, 1) : #6]→[**Complication**: {*.., complicated,...*} (1, n): #8]
  [(REL): _ (>,>) (1, 1) : #7]→[**Complication**: {*.., compliqué,...*} (1, n): #9]
  [(TYPE): {*..., arithmétique, intégral, mental, ...*} (>,<) (1, 1): #10]-
    [**Math_Calculation**:{*..., calcul, ...*} (1, 1): #11]
  [(TYPE): {*..., lithiasique, rénal, urinaire,...*} (>,<) (1, 1): #12]-
    [**Lith_Stone**: {*..., calcul, ...*} (1, 1): #13]
  [(LOC): _ (>,>) (1, 1): #14]-
    [**Kidney**: {*..., rénal,...*} (1, 1): #15]
    [**Urinary Apparatus**: {*..., urinaire,...*} (1, 1): #16]
  [(REL): _ (>,>) (1, 1): #77]→[**Arithmetics**: {*arithmétique, mental*} (1, 1): #78]

This approach realizes a dynamic categorization of ambiguous words with the co-text. It mainly relies on the lexicalization of conceptual relations. Positional indicators and valences are used to restrict compositionality to the closest relevant ambiguous cognitive sign.
With such a piece of semiotic network in the KB, the semantic analysis (SemAn) of *calcul rénal compliqué* gives:

SemAn(*calcul rénal compliqué*)=
  [∅ : *calcul*: #5]-
    [(REL): _ : #7]→[**Complication**: *compliqué*: #9]
    [(TYPE): *rénal*: #11]→[**Lith_Stone**: *calcul*: #12]
    [(LOC): _ : #14]→[**Kidney**: *rénal*: #15]

The conceptual graph corresponding to the above semiotic graph is CG_2:

[∅]-
  (REL)→[**Complication**]

(TYPE)→[**Lith_Stone**]
(LOC)→[**Kidney**]

By type contraction, CG$_2$ gives CG$_3$:

[**Lith_Stone**]-
(REL)→[**Complication**]
(LOC)→[**Kidney**]

3.2 The Semantic of Articles

Another example of the use of the void type can be given with a first simplified cognitive model for the semantic of articles. Articles denote the existence and the count of the noun group they determine. According to the CG model, such a semantic can be supported for the French articles by the following piece of semiotic network, completed below with other refining concepts:

[∅ : {*un, une, des, le, la, les*} (1, n): #17]-
[(quantified as): $ (=, =) (1, 1): #18]-
[⋆ : {*un, une*} (1, 0): #19]
[# : {*le, la*} (1, 0): #20]
[**{⋆}** : {*des*} (1, 0): #21]
[**{#}** : {*les*} (1, 0): #22]
[(has_count): $ (=, =) (1, 1): #23]-
[**@1**: {*un, une, le, la*} (1, 0): #24]
[**Many** : {*les, des*} (1, 0): #25]
[(TYPE): _ (>,>) (1, 1): #26]-
[**Surgeon**: {*..., chirurgien, orthopédiste...*} (1, 1): #27]
[**Sick_Person**: {*..., malade,...*}(1, 1): #28]
[∅ : {*..., calcul,...*} (n, n): #5]-...

The semantic of articles is more ambiguous and contextual, requiring a more sophisticated cognitive model if we want to distinguish existential and universal quantification (e.g. *the kidney is an organ* vs *the kidney has a tumoral lesion*). Nevertheless, with the above extended but simple semiotic network, we obtain:

SemAn(*un calcul compliqué*)=
[∅ : *un*: #17]-
[(quantified as): $: #18]→[⋆ : *un*: #19]
[(has_count): $: #23]→[**@1** : *un*: #24]
[(TYPE): _ : #26]→[∅ : *calcul*: #5]-
[(REL): _ : #7]→[**Complication**: *compliqué*: #9]

Such a semiotic scheme considers that the French article *un* denotes intrinsically a void type entity ∅ quantified as existent (⋆) and numbered as one unit. With such an approach, the denotation of *un* is not limited to the atomic concept [@**1**] but related to a molecule of "meanings" comprising a void typed bracket, its quantifier and its cardinality.

The above semiotic graph can be easily translated into a CG$_4$:

```
[ ∅ ]-
   (quantified as)→[ ⋆ ]
   (has_count)→[@1]
   (TYPE)→[ ∅ ]→(REL)→[Complication]
```

By type contraction, we obtain CG$_5$:

```
[[ ∅ ]→(REL)→[Complication]]-
   (quantified as)→[ ⋆ ]
   (has_count)→[@1]
```

By type referent and count contraction, we get CG$_6$:

```
[[ ∅ ]→(REL)→[Complication]: ⋆ @1 ]
```

The type of *calcul* (calculus) remains here above unspecified, while it is refined by *compliqué* (complicated). This figure permits to refine an untyped concept.

In contrast, because *chirurgien* denotes without ambiguity **Surgeon,** the void cognitive sign triggered by the article *le* will be directly related to its type:

SemAn(*le chirurgien*)=

```
[ ∅ : le: #17]-
   [(quantified as): $: #18]→[ # : le: #20]
   [(has_count): $: #23]→[@1: le: #24]
   [(TYPE): _ : #26]→[Surgeon: chirurgien: #27]
```

This semiotic graph can be turned into CG$_7$:

```
[ ∅ ]-
   (quantified as)→[ # ]
   (has_count)→[@1]
   (TYPE)→[Surgeon]
```

By type, referent and count contraction, we obtain an embedded structure CG$_8$:

```
[[Surgeon]: # @1 ]
```

3.3 Dynamic Allocation of Roles

The bi-transitive verb *opérer* in French denotes an action conform to some method or plan, whose type is further specified by the context as a military deployment, a surgical procedure or a mathematical operation. The context can be related to the locution situation. It can also be simply comprised in the co-text, especially the direct object complement noun group. We found that a conjugated verb in a text is a sufficient conditions to trigger complex concepts such as **Proposition** and **Situation** [20]. It also brings temporal connotations and indications on roles played by the subject and the direct object complement.

```
[Proposition: {..., opère,...} (1, 1): #29]-
 [(DESCR): $ (=, =) (1, 1): #30]→[Situation: {..., opère, ...} (1, n): #31]-
  [(BEGN): $ (=, =) (1, 1): #32]→[Past: {..., opère,...} (1, 0): #33]
  [(COURS): $ (=, =) (1, 1): #34]→[Present: {..., opère,...} (1, 0): #35]
  [(LOC): $ (=, =) (1, 1): #36]→[Place: {..., opère,...} (1, n): #38]
  [(DESCR): $ (=, =) (1, 1): #40]→[Agent: {..., opère, ...} (1, n): #41]-
   [(DESCR): _ (<,<) (1, 1): #42]-
    [ ∅ : {..., un, ...} (1, n): #17]...
    [ ∅ : {il, elle, on, ils, elles} (1, 1): #87]
   [(AGT-1): $ (=, =) (1, 1): #43]→[Action: {..., opère, ...} (1, n): #44]-
    [(TYPE): {..., un, ...} (>,>) (1, 1) : #45]-
     [Surgery: {..., malade,...} (1, 1): #46]
     [∅ : {..., calcul,...} (1, 1): #47]→[(TYPE): _ (>,>) (1, 1): #48]-
      [Surgery: {..., rénal,...} (1, 1): #49]
      [Thought: {...arithmétique, mental,...} (1, 1): #50]
    [(PTCT): $ (=, =) (1, 1): #51]→[Participant: {..., opère,...} (1, n): #52]-
     [(TYPE): {..., un, ...} (>,>) (1, 1): #53]-
      [Beneficiary: {..., malade,...} (1, n): #54]
      [∅ : {..., calcul,...} (1, 1): #55]→[(TYPE): _ (>,>) (1, 1): #56]-
       [Patient: {..., rénal,...} (1, 1): #57]
       [Theme: {...arithmétique, mental,...} (1, 1): #58]
    [(DESCR): _ (>,>) (1, 1) : #59]→[ ∅ : {..., un, ...} (1, n): #17]...
```

This piece of semiotic network comprises a scheme shared by any lexia responding to morphologically unambiguous conjugated forms of French verbs. This example illustrates the place we assign to morphology and syntax in the semantic lexicon. Any lexia that is categorized as a verb without ambiguity will be also a valid lexical entry for **Proposition** and **Situation** that will auto-combine to each other by [(DESCR): $: #30]:

```
[Proposition: {..., opère, opèrera, opérait, ...}(1, 1): #29]
[Situation: {..., opère, opèrera, opérait, ...}(1, n): #31]
```

The above semiotic scheme assumes that the action and its participants share the same time and the same location within **Situation** that embeds them by [(DESCR): $: #88]. With such an approach, whose robustness needs to be confirmed, time or location adverbs and circumstantial complements can be related either to the **Situation** or to the **Action**. This feature might be useful for the semantic of sentences such as: *he speaks in the street* vs *he speaks in the microphone*. Any conjugated verb denoting an action with a beginning in the past will be also a valid lexical entry for **Past** and will auto-combine with **Situation** by [(BEGN): $: #32].

We can also point out that the verb *opère* is per se categorized as an **Action** whose subtype will be directly specified as **Surgery** if the direct object is a sick-person in a sentence like *il opère un malade (he operates on a patient).* If the object complement is an ambiguous term, such as *calcul,* the subtype of **Action** will be indirectly specified through *calcul* and its relational adjectives or complement noun group found in the co-text.

SemAn(*il opère un malade*)=

```
[Proposition: opère: #29]-
  [(DESCR): $: #30]→[Situation: opère: #31]-
    [(BEGN): $: #32]→[Past: opère: #33]
    [(COURS): $: #34]→[Present: opère: #35]
    [(LOC): $: #36]→[Place: opère: #38]
  [(DESCR): $: #40]→[Agent: opère: #41]-
    [(DESCR): _ : #42]→[ ∅ : il: #87]
    [(AGT-1): $: #43]→[Action: opère: #44]-
      [(TYPE): un: #45]→[Surgery: malade: #46]
      [(PTCT): $: #51]→[Participant: opère: #52]-
        [(TYPE): un: #53]→[Beneficiary: malade: #54]
        [(DESCR): _ : #59]→[ ∅ : un: #17]-
          [(TYPE): _ : #26]→[Sick_Person: malade: #28]
          [(quantified as): $: #18]→[ ★ : un: #19]
          [(has_count): $: #23]→[@1 : un: #24]
```

Notice here that the French pronoun *il,* labeled with a void type, could also be specified according to the co-text. Last, we can see in the previous piece of semiotic network that a quite generic role **Participant** is given to the direct object. This role will be categorized as a **Beneficiary**, a **Theme** or a **Patient** according to the co-text by over-specification of the generic role **Participant** by a more specific one. Our richer semiotic network, which is also the Knowledge Base used during the processing of texts, permits us to perform the contextual analysis of sentence (2):

SemAn(*le chirurgien opère un calcul arithmétique*)=

```
[Proposition: opère: #29]-
  [(DESCR): $: #30]→[Situation: opère: #31]-
    [(BEGN): $: #32]→[Past: opère: #33]
    [(COURS): $: #34]→[Present: opère: #35]
    [(LOC): $: #36]→[Place: opère: #38]
  [(DESCR): $: #40]→[Agent: opère: #41]-
    [(DESCR): _ : #42]→[ ∅ : le: #17]-
      [(quantified as): $: #18]→[ # : le: #20]
      [(has_count): $: #23]→[@1: le: #24]
      [(TYPE): _ : #26]→[Surgeon: chirurgien: #27]
    [(AGT-1): $: #43]→[Action: opère: #44]-
      [(DESCR): un: #45]→[ ∅ : calcul: #47]-
        [(TYPE): _ : #48]→[Thought: arithmétique: #50]
      [(PTCT): $: #51]→[Participant: opère: #52]-
        [(TYPE): un: #53]→[∅ : calcul: #55]-
      [(TYPE): _ : #56]→[Theme: arithmétique: #58]
    [(DESCR): _ : #59]→[ ∅ : un: #17]-
      [(quantified as): $: #18]→[ ⋆ : un: #19]
      [(has_count): $: #23]→[@1 : un: #24]
      [(TYPE): _ : #26]→[ ∅ : calcul: #5]-
        [(TYPE): arithmétique: #10]→[Math_Calculus: calcul: #11]
        [(REL): _ : #77]→[Arithmetics: arithmétique: #78]
```

The corresponding CG is:

```
[Proposition]-
  (DESCR)→[Situation]-
    (BEGN)→[Past]
    (COURS)→[Present]
    (LOC)→[Place]
    (DESCR)→[Agent]-
      (DESCR)→[ ∅ ]-
        (quantified as)→[ # ]
        (has_count)→[@1]
        (TYPE)→[Surgeon]
      (AGT-1)→[ Action]-
        (DESCR)→[ ∅ ]→(TYPE)→[Thought]
        (PTCT)→[Participant]-
          (TYPE)→[∅ ]→(TYPE)→[Theme]
          (DESCR)→[ ∅ ]-
            (quantified as)→[ ⋆ ]
            (has_count)→[@1]
            (TYPE)→[ ∅ ]→(TYPE)→[Math_Calculus]
            (REL)→[Arithmetics]
```

By descriptor, type and referent contraction, we obtain the final embedded CG:

```
[Proposition: [Situation:[Agent:[Surgeon: # @1]]-
  (AGT-1)→[Action: [Thought]]-
    (PTCT)→[Theme: [[Math_Calculus]→(REL)→[Arithmetics]: ⋆ @1]]]-
      (LOC)→[Place]
      (BEGN)→[Past]
      (COURS)→[Present]]
```

The **Proposition** describes a **Situation** that has a beginning in the **Past,** a course in the **Present** and a location which is an unspecified **Place**. The **Situation** describes an **Agent**: a definite surgeon, who performs some **Thought** whose theme is a **Math_Calculus** related to **Arithmetics**.

4 Conclusion

Lexical semantic disambiguation consists in the selection of the relevant signification of a term within a predefined list of meanings according to the context [20]. Disambiguation is a treatment for polysemy [21], which is a very generalized phenomenon, even in so-called technical and scientific languages

[22]. Our approach, introducing a void type, supports the dynamic construction of meaning [16], not only by compositionality and conceptual refinement [23] on the syntagmatic axis, but also by dynamic categorization of ambiguous lexia using a type contraction operation. We are able simultaneously to preserve an ambiguity and to refine it with other recognized concepts. A void type is used to make no hypothesis on the meaning of a given lexia. The over-specification of roles such as participant with the co-text illustrates that one can also allocate a very generic type to a given lexia, and refined it with the co-text. The use of a void type also realizes the creation of a "conceptual context" (e.g. contexts as defined in the CG model and not the enunciative context) that can embed a CG as a descriptor, that can be typed with a CG and that can be quantified according to articles and to the co-text. The question of the robustness and the tractability of such an approach on large corpus requires further works. Nevertheless, along our results on medical terminologies integration, we found that the definition of relational cognitive signs signatures - the support of the compositionality - was leading to build kind of cognitive grammars in labeling words with types [17]. Because of the possible use of an auto-combining lexia, a given cognitive sign, even a large part of a given semiotic scheme can be shared by many words. The re-usability of the cognitive grammars appears promising in terms of tractability.

Our approach does not wait pragmatic analysis for desambiguation. We do not resolve ambiguities at the conceptual level according to some inference rules. Other approaches based on morpho-dynamic models as designed in [24] require complex connexionnist implementations. They have been evaluated on limited examples. Disambiguation in our approach is related to the compositionality principles that underlies our semiotic model. It relies also on the lexicalization of conceptual relation using the disambiguating words found in the co-text. It implies thus to revisit the way we allocate relational and conceptual type labels to lexia: the objective is to get efficient and reliable cognitive grammar rules driving the composition of atoms of sens into the relevant meaningfull molecule. Last, this paper focuses on the first step of MLU: the transformation of a NL input into some deep semantic structure. Our semiotic graphs are easily converted to CGs. CG model provides a set of knowledge treatments and inference procedures that do support reasoning or information retrieval that are the only mean to check if the model really fits to the linguistic data.

References

1. Rassinoux AM, Baud RH, Scherrer JR. Conceptual graphs model extension for knowledge representation of medical texts. in Lun, Degoulet , Piemme, Reinhoff Eds. MEDINFO 92. Amsterdam: North-Holland Pub Comp 1992 ; 1368-74.
2. Zweigenbaum P., Bachimont B., Bouaud J., Charlet J., Boisvieux JF. A multilingual architecture for building a normalised conceptual representation from medical language. in Gardner RM ed. Proc Nineteenth Ann Symp Computer Applications in Medical Care, New Orleans, JAMIA supp. 1995; 357-61.

3. Smart JF, Roux M.: A model for medical knowledge representation application to the analysis of descriptive pathology reports. Methods Inf Med 1995; 34: 352-60.
4. Baud RH, Rassinoux AM, Scherrer JR. Knowledge Representation of Discharge Summaries. In Proc of 3rd Euro Conf on Art Intel in Medicine. Marseilles, Springer-Verlag, Fieschi M Ed. 1991; 173-84.
5. Cimino JJ, Barnett GO. Automatic Knowledge Acquisition from Medline. Meth. Inform. Med. 1993; 32: 120-130.
6. Sneiderman C, Rindflesh T, Aronson A. Finding in the Findings: Identification of findings in Medical litterature using restricted Natural Language Processing. Proceedings of the 1996 AMIA fall symposium. Philadelphia. Henley & Belfus. 1996; 239-43.
7. Lovis C, Michel PA, Baud RH, Scherrer J-R . Use of a Conceptual Semi-Automatic ICD-9 Encoding System in an Hospital Environment. Proceedings of 5th Conference on Artificial Intelligence in Medicine in Europe, Pavia, Springer 1995; 331-39.
8. Delamarre D, Burgun A, Seka LP, Le Beux P. Automated coding system of patient discharge summaries using conceptual graphs. Meth Inform Med, 1995; 34: 345-51.
9. Nangle B, Keane MT. Effective retrieval in Hospital Information Systems : the use of context in answering queries to Patient Discharge Summaries. Artif Intell Med 1994; 6 (3): 207-27.
10. Zweigenbaum P, Grabar N. Automatic Acquisition of Morphological Knowledge for Medical Language Processing. in LNAI 1620, 1999; 416-20.
11. Danlos L, Véronis J. Un demi-siècle de traitement automatique des langues: PrŽsentation. TAL. 1997; 38 (2): 3-6.
12. Wilcox AB, Hripcsak G. TheRole of Domain Knowledge in Automating Medical Text Report Classification. J Am Med Inform Assoc, 2003; 10: 330-338.
13. Kossovsky MP, Sarasin FP, Bolla F, Gaspoz JM, Borst F. Distinction between planned and unplanned readmissions following discharge from a department of internal medicine. Methods Inf Med, 1999; 1: 140-143.
14. Hobbs J R. The Generic Information Extraction System. in Proc of the Fifth Message Understanding Conference. Kaufmann. San Francisco. 1993; 87-91.
15. Wittgenstein L. Philosophical Investigations. B Blackwell, Oxford. 1953.
16. Pustejovsky J. The Generative Lexicon. MIT press, Cambridge, MA. 1995.
17. Jacquelinet C, Burgun A, Delamarre D et Al. Developing the ontological foundations of a terminological system for end-stage diseases, organ failure, dialysis and transplantation. Int J Med Inf. 2003; 70(2-3): 317-28.
18. Jacquelinet C. Semiotics and Lexical Semantic Principles for Medical Language Understanding with CG. in B Ganter, A de Moor (Eds), Using Conceptual Structures – Contributions to ICCS, Aachen, 2003; 97-110.
19. Sowa JF. Conceptual Structures: Information Processing in Mind and Machine. Systems Programming Series. Addison Wesley; 1984.
20. Ide N, Veronis J. Introduction to the special Issue on Word Sense Disambiguation: the State of the Art. in P. Steffens Eds, Machine Translation and the Lexicon, Berlin, 1998; 24 (1).
21. Brun C, Jacquemin B, Segond F.: Exploitation de dictionnaires électroniques pour la désambiguïsation sémantique lexicale. TAL, 2001; 42 (3): 667-690.
22. Chute CG, Band RH, Cimino JJ, Patel VL, Rector AL. Coding and Language Processing. Meth Inf Med, 1998; 37: 311-375.
23. Compositionnalité. in TAL, 1998; 39 (1), special issue.
24. Victori B, Fuchs C. La polysémie: construction dynamique du sens. Hermes, Paris, 1996.

Clustering of Conceptual Graphs with Sparse Data

Jean-Gabriel Ganascia and Julien Velcin

LIP6, Université Paris VI
8 rue du Capitaine Scott
75015 PARIS
FRANCE
{jean-gabriel.ganascia,julien.velcin}@lip6.fr

Abstract. This paper gives a theoretical framework for clustering a set of conceptual graphs characterized by sparse descriptions. The formed clusters are named in an intelligible manner through the concept of stereotype, based on the notion of default generalization. The cognitive model we propose relies on sets of stereotypes and makes it possible to save data in a structured memory.

Introduction

This paper proposes a formal approach for the clustering of conceptual graphs characterized by sparse descriptions. According to R. Michalski [1], conceptual clustering is a learning task that takes a set of object descriptions as input and creates a classification scheme. In addition, not only does conceptual clustering have to cluster facts, but it also has to name these clusters. It is crucial to build intelligible descriptions of all generated clusters that can summarize data in an understandable manner. The facts we have considered are rough descriptions of situations taken, for instance, from newspaper articles. These data are not given in an array where all the predefined attributes have been given values, but are characterized by heterogeneous descriptions. In order to deal with such types of data, it was necessary to extend clustering techniques to sparse descriptions.

The names of the clusters, also called labels, are usually calculated either using classical generalization (cf. UNIMEM [2]) or probabilistic generalization (cf. COBWEB [3]). This means that these names are derived from properties common to the facts belonging to the clusters, but in the case of sparse data it fails because there are very few common descriptors. This paper proposes a new approach that could deal with such cases, by building cluster names made up of all the non-contradictory features belonging to the clusters. These names are called stereotypes, by analogy with the concept of prototypes, which are based on a new notion of generalization, default generalization. This concept of stereotypes is well adapted to sets of facts described using sparse descriptions. We therefore consider a cognitive model that structures memory from facts into sets of stereotypes.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 156–169, 2004.

Section 1 presents this cognitive model. The first subsection introduces the notion of default generalization – and shows how it is linked to the compatibility relation in the conceptual graph formalism – which is the basis of our model. The second subsection describes the concept of stereotypes and compares it with the well-known concept of prototypes. Section 2 introduces tools and strategies to build the sets of stereotypes, and section 3 examines the question of validation and suggests an application.

1 The Cognitive Model

Memory has to be structured in order to retrieve old information without registering all the facts and to infer new knowledge from these recorded data. The notion of default generalization will be presented first, followed by the concept of stereotypes.

1.1 Notion of Default Generalization

The facts we have considered have a large number of missing values, as a result of which much of the information has to be guessed in order to perform the clustering. R. Reiter presents in [4] a logic for default reasoning where default rules enable new formulas to be inferred if the hypotheses are not inconsistent. The following rule translates a possible reasoning for the end of the 19th in France:

`politician(X) ∧ introducedAbroad(X) : ¬ diplomat(X) / traitor(X)`

This statement means that a politician who is introduced abroad is a traitor towards his own country if we cannot prove that he is a diplomat. Here, a stereotype stored in the memory will complete a fact if this fact has no contradictory features and is not more similar to another stereotype. The second condition can be checked using the dissimilarities measure and will be seen in the next section. The first and most important condition is verified if the stereotype *generalizes* the fact *by default.* This means that the fact can be completed in such a way that it can now be generalized by the stereotype.

Let us consider the graph g associated with a fact in which there is a very large number of missing values. The missing values can be guessed and completed to obtain a more specific graph g_S. Let us note that we follow here the notations given by Sowa in [5] (definition 3.5.1): $g_S \leq g$ means that g_S is a specialization of g and g is a generalization of g_S [1].

Now, let s be one stereotype belonging to the structured memory. If this stereotype is more general than g_S, ie $g_S \leq s$, then it generalizes g *by default.* More formally:

[1] Many other formalizations exist that encompass the notion of generalization, for instance least general generalization [6] or, more recently, Inductive Logic Programming [7]. For the sake of clarity, we will limit ourselves here to just one formal framework, that designed by Sowa.

Definition 1. *Let f be a fact and s a stereotype, both of them represented using conceptual graphs. s* generalizes f by default *if and only if there exists a graph g_S with $g_S \leq f$ and $g_S \leq s$. g_S is therefore a graph formed by the join operator performed on the graphs f and s.*

Property 1. The notion of default generalization is more general than that of classic generalization. Let g and g' be two conceptual graphs. If g generalizes g' then g generalizes g' by default.

Property 2. The default generalization is a symmetrical relation. Let u_1 and u_2 be two conceptual graphs. If u_1 generalizes u_2 by default, then u_2 generalizes u_1 by default too.

Fig. 1 presents the fact *The meal of Jules is composed of steak, red wine, and ends with a cup of very hot coffee* which can be generalized by default by the stereotype *The meal is composed of steak with potatoes and French bread, and ends with a cup of coffee* because the fact can be completed to *The meal of Jules is composed of steak with potatoes and French bread, red wine, and ends with a cup of very hot coffee.* If the stereotype had presented a meal ending with a liqueur, it would not match the fact and so could not generalize it by default.

This notion allows us to propose stereotypes that label sets of facts and not generalize each fact in the ordinary way but using a default reasoning. Fig. 3 shows three facts that could only be generalized in the ordinary way by the graph `[MEAL]` → `(mainly composed of)` → `[DISH]`, which does not characterize the observations satisfactory. Fig. 4 presents a stereotype that can cover these facts with the default generalization notion.

Let us now compare in more detail the concepts of stereotype and prototype.

1.2 Concept of Stereotype

Eleanor Rosch saw the categorization itself as one of the most important issues in cognitive science. She observed that children learned how to classify first in terms of concrete cases rather than through defining features. She therefore introduced the concept of prototype [8] as the ideal member of a category. Members of the same class might share only a few of those features but are closer to the same prototype, and so are grouped together. For example, a robin is closer to the bird prototype than an ostrich, but they are both closer to it than they are to the prototype of a fish, so we call them both birds. However, it takes longer to say an ostrich is a bird than it takes to say a robin is a bird, because the ostrich is further from the prototype.

Sowa defines a prototype in [5] as a typical instance formed by joining one or more schemata. Instead of describing a specific individual, it describes a typical or "average" individual. Fig. 2 shows the example of a prototype for ELEPHANT in linear form.

Many sufficiently complete observations have to be considered in order to construct such an individual. We therefore propose a new concept, stereotype,

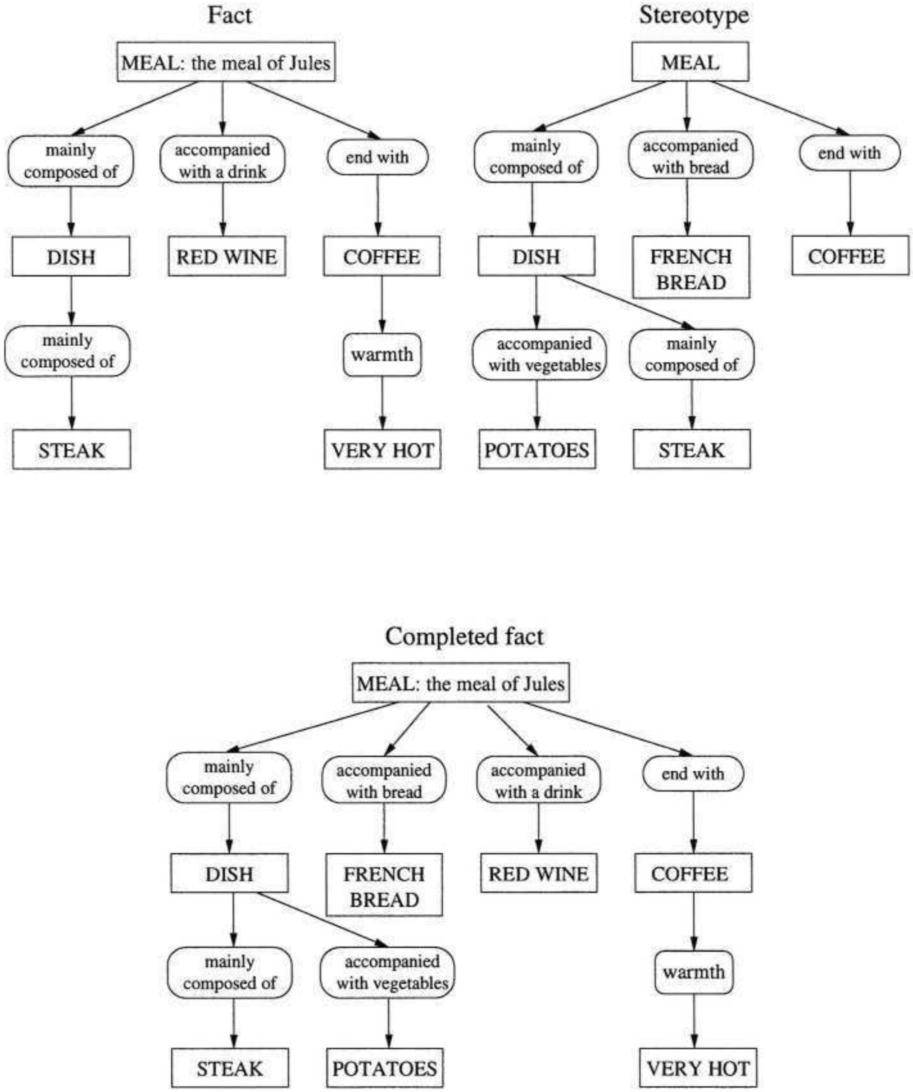
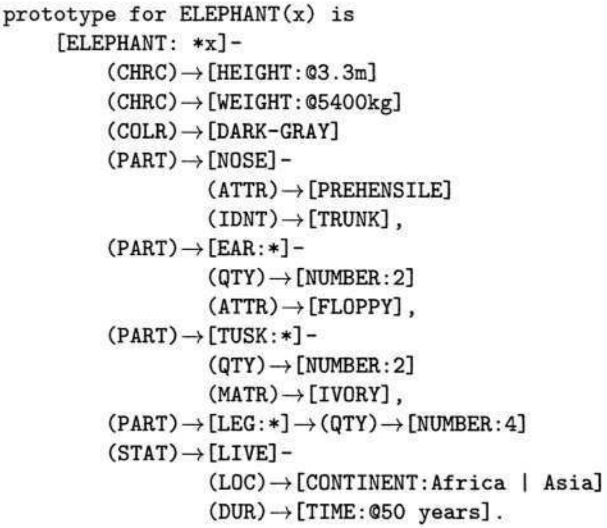


Fig. 1. The stereotype *generalizes by default* the presented fact. The fact below is the result of the join operator.

which is quite close to that of prototype but more adapted to missing values. A stereotype is an imaginary fact that combines features found in the facts covered rather than by performing averages. There is no contradiction between a fact and the related stereotype which covers this fact and may be used to guess missing data. It is because calculating an average is not adapted to a large number of



```
prototype for ELEPHANT(x) is
    [ELEPHANT: *x]-
        (CHRC)→[HEIGHT:@3.3m]
        (CHRC)→[WEIGHT:@5400kg]
        (COLR)→[DARK-GRAY]
        (PART)→[NOSE]-
                (ATTR)→[PREHENSILE]
                (IDNT)→[TRUNK],
        (PART)→[EAR:*]-
                (QTY)→[NUMBER:2]
                (ATTR)→[FLOPPY],
        (PART)→[TUSK:*]-
                (QTY)→[NUMBER:2]
                (MATR)→[IVORY],
        (PART)→[LEG:*]→(QTY)→[NUMBER:4]
        (STAT)→[LIVE]-
                (LOC)→[CONTINENT:Africa | Asia]
                (DUR)→[TIME:@50 years].
```

Fig. 2. A prototype for ELEPHANT.

Fig. 4 represents a stereotype formed from three facts taken from fig. 3 corresponding to French meals. The missing values in these facts can be completed using default reasoning with the corresponding values in the stereotype because there is no contradiction between them. Thus, we can infer for instance that Tom drinks wine or that the Petit family eat French bread with their meal.

2 Construction of the Stereotypes

This section is intended to show a way to automatically structure the memory from a set of facts. The main objective of our work is to find a set of stereotypes that summarize observed situations. The associated graphs have to contain enough information not only to classify new observations but also to make possible the reasoning on those situations.

2.1 The Dissimilarities Measure

We need to establish a measure to calculate the distance between two conceptual graphs, one being a fact to be covered and the other the candidate-stereotype. Previous work deals with graph matching and an interesting method to calculate the similarity between two conceptual graphs is proposed in [9]. However, in the present context, a measure which adds the differences is sufficient.

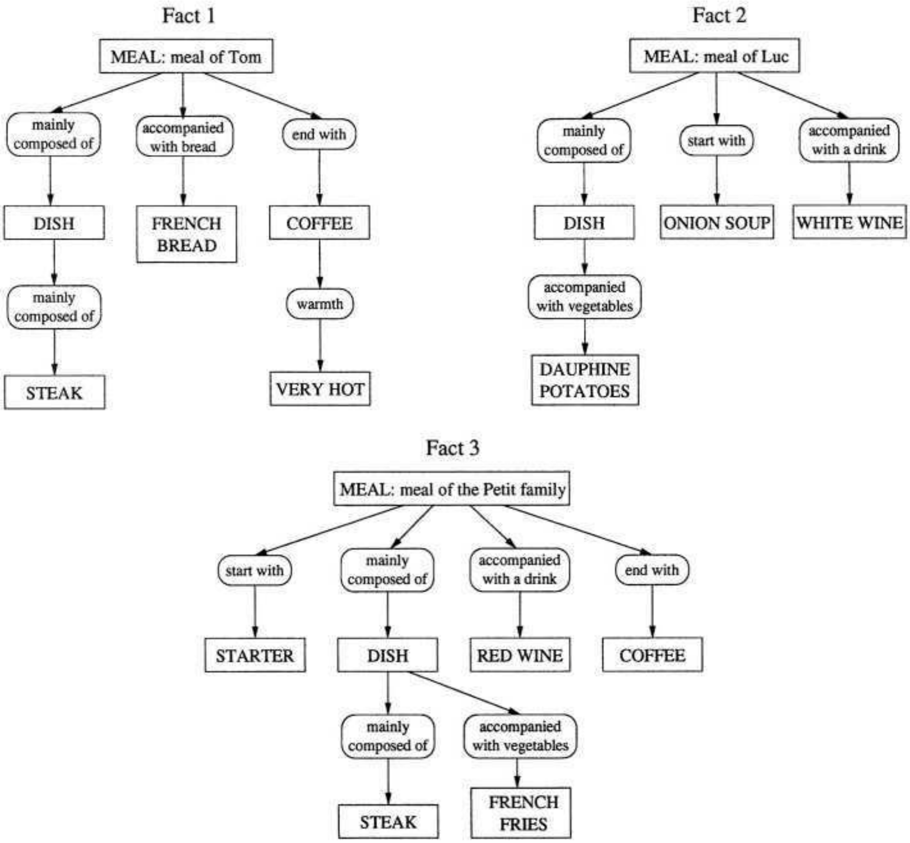


Fig. 3. Three facts corresponding to a French meal.

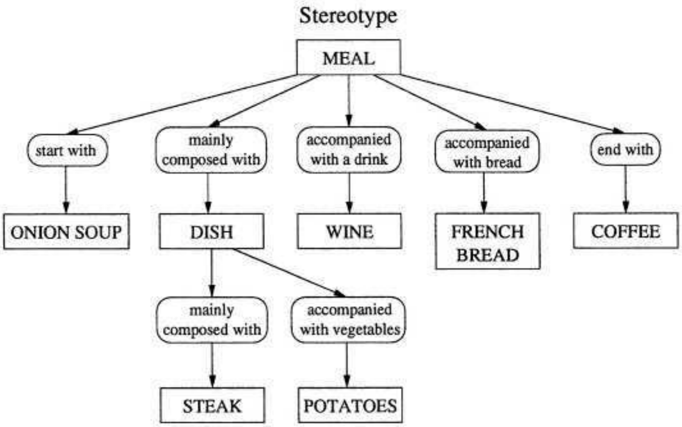


Fig. 4. A stereotype of a French meal.

First let us recall the definition of *compatibility* given in [5]:

Definition 2. *Let conceptual graphs u_1 and u_2 have a common generalization v with projections $\pi_1 : v \rightarrow u_1$ and $\pi_2 : v \rightarrow u_2$. The two projections are said to be* compatible *if for each concept c in v, the following conditions are true:*

1. $type(\pi_1 c) \cap type(\pi_2 c) > \perp$.
2. *The referents of $\pi_1 c$ and $\pi_2 c$ conform to $type(\pi_1 c) \cap type(\pi_2 c)$.*
3. *If $referent(\pi_1 c)$ is the individual marker i, then $referent(\pi_2 c)$ is either i or $*$.*

We now consider that there is always only one least common generalization, i.e. only two projections that are compatible and maximally extended. It is easy to generalize with graphs having several least common generalizations.

We state the following theorem in order to link the notions of compatibility and default generalization:

Theorem 1. *Let conceptual graphs u_1 and u_2 have the least common generalization v with projections $\pi_1 : v \rightarrow u_1$ and $\pi_2 : v \rightarrow u_2$. π_1 and π_2 are compatible if and only if u_1 generalizes u_2 by default.*

Proof. If π_1 and π_2 are compatible then there exists a common specialization w of u_1 and u_2 (cf. theorem 3.5.7 [5]). According to definition 1, u_1 generalizes u_2 by default. Reciprocally, if u_1 generalizes u_2 by default then there exists a common specialization w. Suppose that π_1 and π_2 are not compatible. There therefore exists at least one concept in v with $type(\pi_1 c) \cap type(\pi_2 c) = \perp$, or with the referent of $\pi_1 c$ or $\pi_2 c$ not conform to $type(\pi_1 c) \cap type(\pi_2 c)$, or with $referent(\pi_1 c) = i$ and $referent(\pi_2 c) = j$, $i \neq j$. These three cases are absurd because they contradict the construction of w. Therefore, π_1 and π_2 are compatible.

Consider now the measure M_D counting the dissimilarities between two graphs u_1 and u_2. Let v be the least common generalization graph with projections $\pi_1 : v \rightarrow u_1$ and $\pi_2 : v \rightarrow u_2$. If π_1 and π_2 are not compatible then the measure $M_D(u_1, u_2)$ is fixed by convention with an infinite value noted M_∞ because one graph can't be generalized by default by the second one (cf. theorem 1). Otherwise $M_D(u_1, u_2)$ counts all the differences between the concepts and relations of u_1 and those of u_2. The measure is thus defined:

Definition 3. *Let conceptual graphs u_1 and u_2 have the least common generalization v with projections $\pi_1 : v \rightarrow u_1$ and $\pi_2 : v \rightarrow u_2$. The* measure of dissimilarities *$M_D(u_1, u_2)$ is equal to:*

1. *M_∞ if π_1 and π_2 are not compatible.*
2. *$C + T(u_1) + T(u_2)$ otherwise, where:*
   - $C = |\{concept\ c \in v / type(\pi_1 c) \neq type(\pi_2 c)\ or\ referent(\pi_1 c) \neq referent(\pi_2 c)\}|$.
   - *$T(u) = card(u) - card(v)$; $card(g)$ corresponds to the number of nodes (concepts and relations) of graph g.*

This measure presents the following properties:

Property 3. For any conceptual graph u, $M_D(u, u) = 0$.

Property 4. For any conceptuals graphs u and v, $M_D(u, v) = M_D(v, u)$.

2.2 Relative Cover

We structure the memory into a set of stereotypes that summarizes the observations. Each fact, thanks to the measure defined above, is associated with the closest stereotype and completed with its description. *Relative cover* allows this calculation :

Definition 4. *The fact* f *is* covered *by the stereotype* s relative to *a set of stereotypes* $\{s_1, ..., s_r\}$ *if and only if:*

1. $\exists i,\ 1 \leq i \leq r/s = s_i$.
2. $M_D(f, s) \neq M_\infty$.
3. $\forall k \neq i, 1 \leq k \leq r, M_D(f, s) < M_D(f, s_k)$.

Fig. 5 shows a set of three stereotypes that may be read s_1 = *The meal is accompanied by French bread with cheese and red wine from Bordeaux,* s_2 = *The meal is composed of steak with potatoes and beer* and s_3 = *The meal is composed of poultry with salad and mineral water.* Fig. 6 presents two facts to be classified within these stereotypes: f_1 = *The meal of Bob is composed of cheese and wine* and f_2 = *The meal of Léa is composed of salad and ends with a coffee without sugar.*

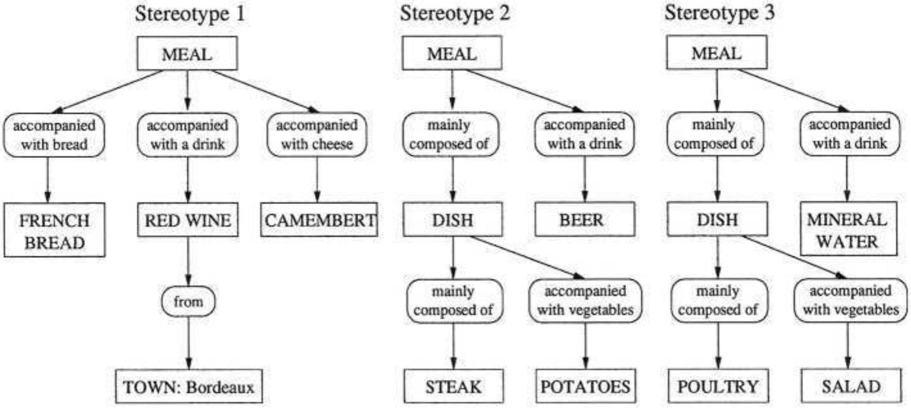


Fig. 5. Three stereotypes of a French meal.

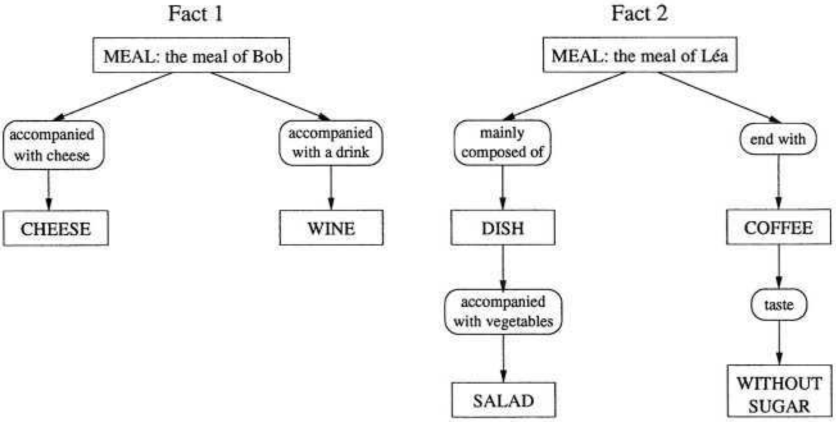


Fig. 6. Two facts to be classified.

Let us first consider the fact f_1 and the stereotype s_1. The common generalization v is *The meal is composed of cheese with wine.* $M_D(f_1, s_1)$ is therefore equal to 3 + (5 – 5) + (9 – 5) = 7. The stereotype s_2 gives $M_D(f_1, s_2) = 2+(5-3)+(9-3) = 10$. Finally, $M_D(f_1, s_3)$ is equal to M_∞ because *MINERAL WATER* $\cap$ *WINE* $=\perp$. f_1 is therefore covered by s_1 relative to s_1, s_2, s_3. This means that f_1 is associated with the cluster labeled with the stereotype s_1.

The same calculation on the second fact f_2 gives the measures $M_D(f_2, s_1) = 17$, $M_D(f_2, s_2) = M_\infty$ and $M_D(f_2, s_3) = 9$. The fact is therefore covered by s_3 relative to s_1, s_2, s_3 and classified in the cluster labeled with s_3. If no stereotype covers a fact relative to the others then the fact is classified in a "garbage" cluster.

This is the case if no stereotype generalizes it by default or if two stereotypes get the same best value for M_D, because one fact cannot be covered by two different stereotypes.

2.3 Search for the Best Set of Stereotypes

The aim is to find the best set of stereotypes given a set of considered facts. To decide which set of conceptual graphs is the best, the definition of an evaluation function is needed.

The first function considered uses probabilities. It is very similar to the *Category Utility* measure (see [10]) which is used in the COBWEB system [3] to evaluate good partitions. But this measure is not really appropriate for sparse descriptions and uses the Attributes/Values formalism. Moreover, runtime cost was rather high.

What we propose here is a more classical way using the definition of an evaluation function based on the dissimilarities measure M_D. This function is minimized by the best set of stereotypes.

Definition 5. *Let F be the set offacts to be classified, S the set of stereotypes to be evaluated and* **s** *the function that associates the fact* f *of F to the stereotype* $s(f)$ *of S such that* f *is coveredby* $s(f)$ *relative to S. The cost function* **h** *is defined below:*

$$h(S) = \sum_{f \in F} M_D(f, s(f))$$

There are several methods for exploring the search space. One is incremental and very similar to that used by Fisher in [3]. It starts from an empty set with no stereotypes, considers at each step a new fact to be covered and updates the set with some operators. There could be one operator which creates a new stereotype equal to the considered fact and another which modifies an existing stereotype to cover it. The "merge" operator is a little bit more difficult to implement, but the main difficulty lies in the "split" operator, especially when there are conceptual structures. The search in COBWEB is like a "hill-climbing" strategy and its robustness is largely due to these two specific operators. A more general approach is therefore certainly preferable.

The second option is to search for the best set of stereotypes using a "tabu" strategy. A neighborhood is calculated from a current solution with the assistance of permitted movements. These movements can be of low influence (add a concept to one stereotype, restrict a concept within another) and high influence (add or retract one stereotype of the set). The search uses short and long-term memory to avoid loops and to intelligently explore the search space.

3 Results

This section begins by examining the question of validation and goes on to present an application.

3.1 Validation

We have considered two ways to validate the proposed model.

The first one relies on real cases using knowledge from the domain studied. The experiments deal with a major issue which was of historic importance in France and so much has been written about it (cf. section 3.2). Articles from newspapers about this affair are very sparse and so fit perfectly into our cognitive model. We have to confirm that the stereotypes that have been built correspond well to the mental representations of the affair, as projected in the media. Preliminary results in the Attributes/Values formalism are promising and can be confronted to the knowledge of the domain provided by books and experts.

The second one gives a more formal and systematic method with the elaboration of artificial training sets. We begin with a set of "complete" facts, i.e. using all possible concepts and relations. Thus, a fact can have identical features with the others, compatible features, or even contradictory features. Each fact is then duplicated several times in order to form a set of perfectly coherent clusters of different sizes. Next, sparse descriptions are simulated by removing part of the data: certain facts are replaced randomly by more general ones. Finally, the clustering algorithm is performed on these new data in order to produce a set of stereotypes. The original considered facts were retrieved. We have also changed the proportion of missing values in order to test the robustness of this method.

Fig. 7 shows a graph which uses the Attributes/Values formalism to give the classification error rate with respect to the missing data rate:

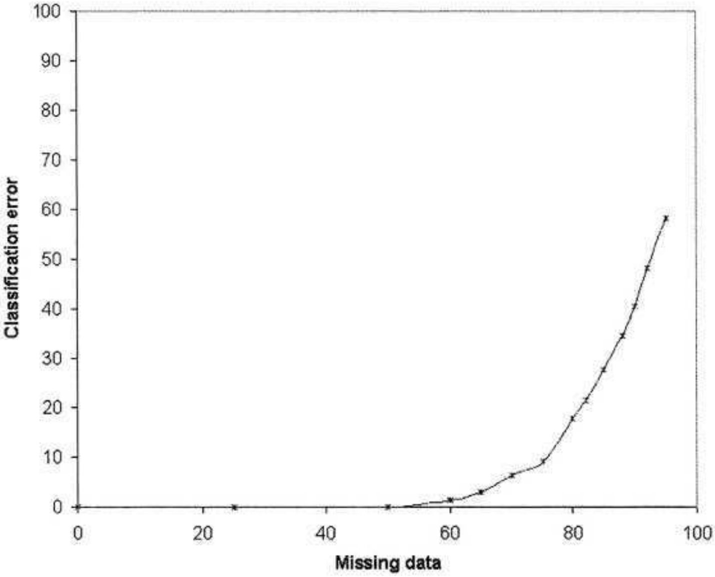


Fig. 7. Classification error using artificial training sets.

These experiments were carried out on three initial random facts until twenty runs in order to calculate averages. Each fact was duplicated fifty times. The classification error rate was counted according to the specific rate of removed descriptors. Errors are due to the facts which are not covered by the stereotype corresponding to the original fact they come from. These results prove the robustness of the method based on the concept of stereotypes, since the error rate is below 10% with up to 75% of missing data. That means that the three original facts were almost perfectly retrieved. Moreover, the rate does not exceed 28% with 85% of missing data, which is still satisfactory.

3.2 Applications

In today's information world there are many situations where data are so sparse that much interpretation is necessary.

The application we propose deals with an historic event, the famous miscarriage ofjustice known as the Dreyfys Affair which occurred at the end of the 19th century in France. In 1894 Captain Alfred Dreyfus, an officer on the French general staff, was accused of spying for Germany, France's opponent in the previous war. There were many articles about this very complex affair, bringing different views depending on the date, recent events, the newspaper political leanings. Thus, the liberal pro-Dreyfus *Le Siècle* expressed opinions which were diametrically opposed to those of the conservative anti-Dreyfus *L'éclair*. The facts we considered have been taken from these articles and translated into conceptual graphs, in order to build automatically a simplified model of the affair. The objective is to understand the influence of the press on the mental representations during this period.

Type hierarchies including 399 concepts and 174 relations were built for this specific context. In addition, a typical graph was proposed in order to translate the articles into facts. Fig. 8 shows an example of a graph. It represents an article from the newspaper *L'éclair* using the CoGITaNT library implemented by D. Genest and E. Salvat [11]:

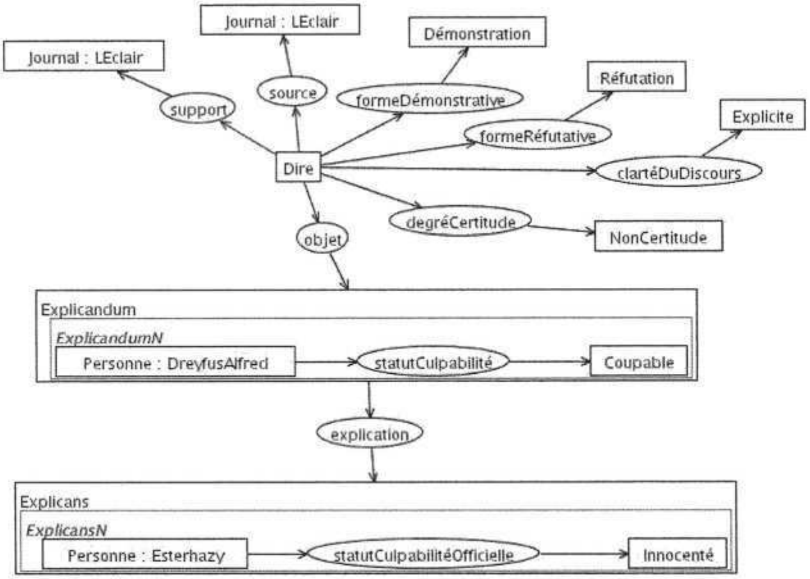


Fig. 8. A conceptual graph which translates a newspaper article.

It could be summarized as follows : *the article taken from the newspaper* L'éclair *explicitly asserts that Alfred Dreyfus is guilty because Esterhazy was proved innocent by the courts.* Once several articles have been translated in this way, stereotypes can be discovered using the methods proposed earlier.

4 Conclusion and Future Research

Flows of information play a key role in today's society. However, the value of information depends on its being interpreted correctly, and implicit knowledge has a considerable influence on this interpretation. This is particularly true of the media, such as newspapers, radio and television, where the information given is always incomplete. In this context we propose a cognitive model based on sets of stereotypes which summarize facts by "guessing" the missing values. Stereotypes are an alternative to prototypes and are more suitable for sparse descriptions. They rely on the notion of default generalization which relaxes constraints and makes possible the manipulation of such descriptions. Descriptions are then completed according to the closest stereotypes, with respect to the dissimilarities measure M_D. The present paper gives definitions of new operators which make this search possible using conceptual graphs formalism.

This work relates to the domain of social representations as introduced by Serge Moscovici in [12]. According to him, social representations are a sort of "common sense" knowledge which aims at inducing behaviors and allows communication between individuals. Social representations can be constructed with the help of sets of stereotypes, and the way these representations change can be studied through the media over different periods and social groups in comparison with such sets. This represents an unexplored way for enriching historical or social analysis.

Finally, this paper is closely related to the work of Herbert Simon [13]. Thus, the construction of internal representations is a fundamental problem in Artificial Intelligence, emerging from the research done on the notion of semantic in the 1960s. These sort of representations bring about desired actions from the large amounts of information gathered from the outside world. Stereotypes and default generalization can be used to summarize these sparse data for which more classical techniques are not appropriate.

Acknowledgments. The authors would particularly like to thank David Genest for his patience and useful advice concerning the CoGITaNT library, Rosalind Greenstein for reading and correcting the manuscript and Anne Remillieux for constructing the type hierarchies and graph examples.

References

1. Michalski, R.S.: Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. In: International Journal of Policy Analysis and Information Systems, number 4 (1980) 219–243.
2. Lebowitz, M.: Experiments with Incremental Concept Formation: UNIMEM. In: Machine Learning, number 2 (1987) 103–138.
3. Fisher, D.H.: Knowledge Acquisition Via Incremental Conceptual Clustering. In: Machine Learning, number 2 (1987) 139–172.

4. Reiter, R.: A logic for default reasoning. In: Artificial Intelligence, number 13 (1980) 81–132.
5. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Company (1984), Massachusetts, The Systems Programming Series.
6. Plotkin, G.D.: A Note on Inductive Generalization. In: Machine Intelligence, number 5. American Elsevier Publishing Company, Inc. (1970) 153–163. Meltzer, Bernard and Michie, Donald, New York.
7. Muggleton, S., De Raedt, L.: Inductive Logic Programming: Theory and Methods. In: Journal of Logic Programming, volume 19/20 (1994) 629–679.
8. Rosch, E.: Cognitive representations of semantic categories. In: Journal of Experimental Psychology: General, number 104 (1975) 192–232.
9. Zhong, J., Zhu H., Li J., Yu Y.: Conceptual Graph Matching for Semantic Search. In: Proceedings of the 10th International Conference on Conceptual Structures: Integration and Interfaces, Spring-Verlag (2002) 92–106.
10. Corter, J.E., Gluck, M.A.: Information, uncertainty, and the utility of categories, In: Proceedings of the Seventh Annual Conference of the Cognitive Science Society. Lawrence Erlbaum Associates (1985) 283–287.
11. Genest, D., Salvat, E.: A platform allowing typed nested graphs: How cogito became cogitant. In: Proceedings of the Sixth International Conference on Conceptual Structures, Springer-Verlag (1998), Berlin.
12. Moscovici, S.: La psychanalyse: son image et son public. PUF (1961), Paris.
13. Simon, H.A., Siklóssy, L. (Eds.): Representation and meaning: experiments with information processing systems. Prentice-Hall, Inc. (1972). Englewood Cliffs, New Jersey.

A Cartesian Closed Category of Approximable Concept Structures

Pascal Hitzler[1]* and Guo-Qiang Zhang[2]**

[1] Artificial Intelligence Institute, Department of Computer Science
Dresden University of Technology, Dresden, Germany
phitzler@inf.tu-dresden.de
www.wv.inf.tu-dresden.de/~pascal/

[2] Department of Electrical Engineering and Computer Science
Case Western Reserve University, Cleveland, Ohio, U.S.A.
gqz@eecs.case.edu; newton@case.edu

Abstract. Infinite contexts and their corresponding lattices are of theoretical and practical interest since they may offer connections with and insights from other mathematical structures which are normally not restricted to the finite cases. In this paper we establish a systematic connection between formal concept analysis and domain theory as a categorical equivalence, enriching the link between the two areas as outlined in [25]. Building on a new notion of *approximable concept* introduced by Zhang and Shen [26], this paper provides an appropriate notion of morphisms on formal contexts and shows that the resulting category is equivalent to (a) the category of complete algebraic lattices and Scott continuous functions, and (b) a category of information systems and approximable mappings. Since the latter categories are cartesian closed, we obtain a cartesian closed category of formal contexts that respects both the context structures as well as the intrinsic notion of approximable concepts at the same time.

1 Introduction

Formal concept analyais (FCA [1]) is a powerful lattice-based tool for symbolic data analysis. In essence, it is based on the extraction of a lattice — called *formal concept lattice* — from a binary relation called *formal context* consisting of a set of objects, a set of attributes, and an incidence relation. The transformation from a two-dimensional incidence table to a lattice structure is a crucial *paradigm shift* from which FCA derives much of its power and versatility as a modeling tool. The concept lattices obtained this way turn out to be exactly the complete lattices, and the particular way in which they structure and represent knowledge is very appealing and natural from the perspective of many scientific disciplines.

* The first author was supported by a Postdoc-fellowship of the German Academic Exchange Service (DAAD); this work was carried out while the author was visiting the second author at the Department of EECS, Case Western Reserve University[2].

** Corresponding author.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 170–185, 2004.

The successful applications of FCA, however, are mainly restricted to finite contexts or finite concept lattices, thus neglecting the full power of the theory. Infinite contexts and their corresponding lattices are of theoretical and practical interest since they may offer connections with and insights from other mathematical structures which are normally not restricted to the finite cases. In this paper we establish a systematic connection between formal concept analysis and domain theory as a categorical equivalence, enriching the link between the two areas as outlined in [25]. Domain theory is a subject of extensive study in theoretical computer science and programming languages. Its basic idea of partial information and successive approximation suggests that for infinite structures to be computationally feasible, items of knowledge or information should either be finitely representable or approximable by such finitely representable items. This idea motivated the introduction of a new notion called *approximable concept,* reported by Zhang and Shen in a separate paper [26]. Approximable concept lattices derived from this new notion are exactly the complete algebraic ones; and every (classical) formal concept is approximable. Furthermore, in cases where the formal contexts are finite, approximable concepts and formal concepts coincide. From a categorical viewpoint, this establishes a relation at the object level as a part of a functor; a main contribution of this paper is the introduction of an appropriate notion of morphisms on formal contexts and the proof that the following three categories are equivalent in the categorical sense [13]:

1. the category of formal contexts and context morphisms **Cxt** introduced here for the first time,
2. the category of complete algebraic lattices and Scott continuous functions, and
3. the category of information systems and approximable mappings **ISys**.

This implies that the category of formal contexts and context morphisms is cartesian closed, and as a result a rich collection of constructions including product and function space is immediately made possible.

Our paper can be viewed as part of a unique research program [8,25,26, 27,19] exploiting the synergies among the following recurring themes in several independently developing and yet somewhat related areas:

- functional dependency $X \rightarrow Y$ in databases,
- association rules $X \Rightarrow Y$ in data-mining,
- consequence relation $\wedge X \models \wedge Y$ in logic,
- entailment relation $X \vdash Y$ in information systems and domains,
- intention-extension duality $Y \subseteq (X)''$ in formal concept analysis.

Note that in both logic as well as in domain theory, X is restricted to finite sets while in databases and data-mining both X and Y are restricted to finite cases due to the pragmatic motivations of the areas. In classical formal concept analysis, there is no size constraint on either X or Y even when the formal contexts are infinite. These potential mismatches (or, the alignment of them) turn out to have important consequences, as will be seen from the rest of this paper.

The rest of the paper is structured as follows. In Section 2 we introduce basic notions from formal concept analysis, domain theory, and category theory. In Section 3, results on approximable concepts are provided. In Section 4 we introduce an appropriate notion called *context morphism* on formal contexts and show that formal contexts with context morphisms constitute a category **Cxt**. In Section 5 we recall the category **ISys** of Scott information systems with trivial consistency predicate. This category is known to be equivalent to the category of complete algebraic lattices with Scott continuous functions. We then introduce the functors which will be used in our equivalence proof, which will be carried out in Section 6. Some categorical constructions in Section 7 complete the technical contributions of the paper. The last section gives some concluding remarks.

2 Background

We review some necessary background in FCA, domain theory, and category theory in order to fix notations.

Formal Contexts and Concepts

Our main reference for formal concept analysis (FCA) is [1]. In places, we will follow the notation used in [26], because it is more convenient for our purposes.

Definition 1. *A* formal context *P is a triple $(P_o, P_a, \models)$, where P_o is a set of* objects, *P_a a set of* attributes, *and $\models$ a binary* satisfaction relation *$\models \subseteq P_o \times P_a$. We also define the following mappings.*

$$\alpha_P = \alpha : 2^{P_o} \to 2^{P_a} \text{ with } X \mapsto \{a \mid \forall x \in X,\, x \models a\}$$
$$\omega_P = \omega : 2^{P_a} \to 2^{P_o} \text{ with } Y \mapsto \{o \mid \forall y \in Y,\, o \models y\}$$

A subset $A \subseteq P_a$ is called an (intent-) concept *if $\alpha(\omega(A)) = A$.*

For readers from the traditional FCA community, it should be helpful to note that P_o corresponds to G, P_a corresponds to M, and $\models$ to I using the standard notation of a context (G, M, I). Also note that $(\quad)' = \alpha$ and $(\quad)' = \omega$ since the standard notation ignores the types of the two operators. One can also regard $(\quad)'$ as, informally, an "infix" notation and α, ω as prefix ones.

The following is a central result for FCA, a proof of which can be found in [1]. Recall that a *complete lattice* is a partial order in which all (possibly infinite) suprema (a.k.a. join) and infima (a.k.a. meet) exist.

Theorem 1 (Wille [24]). *The set of all (intent-) concepts of a formal context P, ordered under subset inclusion, is a complete lattice, called the* concept lattice *of P. Furthermore, every complete lattice $(D, \leq)$ is isomorphic to the concept lattice L of the formal context $(D, D, \leq)$, with isomorphism $\iota : D \to L$ given by $d \mapsto \alpha(\omega(\{d\}))$.*

Domain Theory

Domain theory was introduced by Scott in the late 60s for the denotational semantics of programming languages. It provides a theoretically elegant and practically useful mathematical foundation for the design, definition, and implementation of programming languages, and for systems for the specification and verification of programs. The basic idea of domain theory is partial information and successive approximation, captured in *complete partial orders* (cpos). Functions acting on cpos are those which preserve the limits of directed sets – this is the so-called continuity property. If one thinks of directed sets as an approximating schema for infinite objects, then members of a directed set can be thought of as finite approximations. Continuity makes sure that infinite objects can be approximated by finite computations. An important property of continuous functions is that when ordered in appropriate ways, they form a complete partial order again. Thus a continuous function becomes once again an object in a complete partial order. This seamless and uniform treatment of a higher-order object just as an ordinary object is the hallmark of domain theory.

Let $(D, \sqsubseteq)$ be a partial order. A subset X of D is *directed* if it is non-empty and for each pair of elements $a, b \in X$, there is an upper bound $x \in X$ for $\{a, b\}$. A *complete partial order* (cpo) is a partial order $(D, \sqsubseteq)$ with a least element $(\bot)$ and every directed subset X has a least upper bound (or join) $\bigsqcup X$. A complete lattice is a partial order in which any subset has a join (this implies that any subset will also have a meet – greatest lower bound). Compact elements of a cpo $(D, \sqsubseteq)$ are those inaccessible by directed sets: $a \in D$ is *compact* if for any directed set X of D, $a \sqsubseteq \bigsqcup X$ implies that there exists $x \in X$ with $a \sqsubseteq x$. A cpo is *algebraic* if every element is the join of a directed set of compact elements. A set $X \subseteq D$ is *bounded* if it has an upper bound. A cpo is *bounded complete* if every bounded set has a join. Scott domains are, by definition, bounded complete algebraic cpos.

Category Theory

Category theory provides a unified language for managing *conceptual complexity in mathematics.* The importance of category theory to computer science is manifested in its ability in guiding research to the discovery of categorically natural, but otherwise non-obvious missing entities in a conceptual picture.

Our category-theoretical terminology follows [5]. A *category* $\mathbf{C}$ consists of

(i) a class $|\mathbf{C}|$ of *objects* of the category,
(ii) for all $A, B \in |\mathbf{C}|$, a set $\mathbf{C}(A, B)$ of *morphisms* from A to B,
(iii) for all $A, B, C \in |\mathbf{C}|$, a composition operation
$\circ : \mathbf{C}(B, C) \times \mathbf{C}(A, B) \to \mathbf{C}(A, C)$,
(iv) for all $A \in |\mathbf{C}|$, an *identity morphism* $\mathrm{id}_A \in \mathbf{C}(A, A)$,

such that for all $f \in \mathbf{C}(A, B)$, $g \in \mathbf{C}(B, C)$, $h \in \mathbf{C}(C, D)$, the associativity axiom $h \circ (g \circ f) = (h \circ g) \circ f$ and the identity axioms $\mathrm{id}_B \circ f = f$ and $g \circ \mathrm{id}_B = g$ are satisfied. As usual, we write $f : A \to B$ for morphisms $f \in \mathbf{C}(A, B)$.

A *functor* $\mathbf{F}$ from a category $\mathbf{A}$ to a category $\mathbf{B}$ consists of

(i) a mapping $|\mathbf{A}| \to |\mathbf{B}|$ of objects, where the image of an object $A \in |\mathbf{A}|$ is dentoted by $\mathbf{F}A$,
(ii) for all $A, A' \in |\mathbf{A}|$, a mapping $\mathbf{A}(A, A') \to \mathbf{B}(\mathbf{F}A, \mathbf{F}A')$, where the image of a morphism $f \in \mathbf{A}(A, A')$ is denoted by $\mathbf{F}f$,

such that for all $A, B, C \in |\mathbf{A}|$ and all $f \in \mathbf{A}(A, B)$ and $g \in \mathbf{A}(B, C)$ we have $\mathbf{F}(f \circ g) = \mathbf{F}f \circ \mathbf{F}g$ and $\mathbf{F}\mathrm{id}_A = \mathrm{id}_{\mathbf{F}A}$.

3 Approximable Concepts

The defining property of a formal concept given by the equality $\alpha(\omega(A)) = A$ is computationally feasible for finite contexts, but lends itself for alternative formulation in the infinite case. From a domain-theoretic perspective, a computationally accessible infinite object is one that can be approximated by partial, finitary objects. If we replace "object" by formal concept, and "finitary objects" by finitely generated concepts (i.e., $\alpha(\omega(A))$ for finite A), then we obtain the following definition, introduced in [26].

Definition 2. *Given a set A, let* $\mathsf{Fin}(A)$ *denote the set of finite subsets of A. With notation fixed in Definition 1, a subset* $A \subseteq P_a$ *is called an* approximable (intent-) concept *if for every* $X \in \mathsf{Fin}(A)$*, we have* $\alpha(\omega(X)) \subseteq A$.

As a consequence, every approximable concept A is the limit (i.e. least upper bound) of a directed set of finitely generated concepts below it:

$$A = \bigcup \{\alpha(\omega(X)) \mid X \in \mathsf{Fin}(A)\}.$$

The notion of approximable concept is a natural one from a logical point of view, in that approximable concepts correspond to *theories*. Informally, a (logical) theory is a set of formulas closed under (a predefined notion) of entailment. A basic notion of entailment can be extracted from a context by letting $X \vdash a$ just when $a \in \alpha(\omega(X))$. This has been observed by many researchers and investigated at a more systematic level in [25] by relating it to information systems [18]. The relation $\vdash$ corresponds to an *association rule* in data mining and an instance of *functional dependence* in databases. If we build theories by taking attributes as atomic propositional formulas and the corresponding $\vdash$ as the entailment relation, then theories coincide with approximable concepts. The distinction is that in classical formal concept analysis, an infinite set X is allowed in the entailment $X \vdash a$ while in approximable concept analysis, only finite Xs are allowed. It is well-known in logic that an infinite conjunction p in the antecedent of an entailment $p \to q$ destroys compactness. For readers interested in discussions along this line, we refer to [23] for general and intuitive examples, and to [10] for related hardcore theory.

We now summarize relevant results from [26]. Recall that a complete lattice is called *algebraic* if each of its element is the supremum of the directed set of compact elements below it. Given a complete algebraic lattice $(D, \leq)$, let $\mathsf{K(D)}$ denote the set of all compact elements of D.

Theorem 2 (Zhang and Shen [26]). *For any formal context P, the set of its approximate (intent-) concepts $\mathcal{A}P$ under set-inclusion forms a complete algebraic lattice. Conversely, every complete algebraic lattice $(D, \leq)$ is order-isomorphic to $\mathcal{A}P$, where $P = (D, \mathcal{K}(D), \leq)$. An isomorphism in this case is given by $K \mapsto \sup K$ for any approximable concept K.*

The supremum just mentioned exists since approximable concepts in this case are exactly the ideals (i.e. downward closed directed subsets) of D.

It is easy to see that with respect to finite contexts, approximable concepts are just the standard ones. In the infinite case, although every standard concept is approximable, not all approximable concepts are concepts in the standard sense. This gives the impression that approximable concepts are more general; but interestingly they are less general in terms of the lattices they represent collectively: the corresponding lattices built from approximable concepts are of a restricted kind – the algebraic, complete lattices – instead of complete lattices in general. This again fits the domain-theoretic paradigm in that a general approximating scheme should be part of a computable mathematical structure. We refer to [26] for an example (necessarily infinite) to illustrate the differences.

4 Cxt: A Category of Formal Contexts

We introduce a new notion of morphism on formal contexts.

Definition 3 (context morphism). *Given formal contexts $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$, a* context morphism *$\mathrel{-\!\triangleright}_{PQ} = \mathrel{-\!\triangleright}$ from P to Q is a relation $\mathrel{-\!\triangleright} \subseteq \mathsf{Fin}(P_a) \times \mathsf{Fin}(Q_a)$, such that the following conditions are satisfied for all $X, X', Y_1, Y_2 \in \mathsf{Fin}(P_a)$ and $Y, Y' \in \mathsf{Fin}(Q_a)$:*

(cm1) $\emptyset \mathrel{-\!\triangleright} \emptyset$,
(cm2) $X \mathrel{-\!\triangleright} Y_1$ *and* $X \mathrel{-\!\triangleright} Y_2$ *imply* $X \mathrel{-\!\triangleright} Y_1 \cup Y_2$,
(cm3) $X' \subseteq \alpha_P(\omega_P(X))$ *and* $X' \mathrel{-\!\triangleright} Y'$ *and* $Y \subseteq \alpha_Q(\omega_Q(Y'))$ *imply* $X \mathrel{-\!\triangleright} Y$.

We give some intuition about this notion of morphism. We think of sets of attributes as carrying knowledge, or information, and morphisms from P to Q relate this knowledge in the sense that some knowledge in P implies some knowledge in Q. So $X \mathrel{-\!\triangleright} Y$ should be read as: "If at least X is known, then also at least Y is known." Conditions (cm1) and (cm2) are easily understood from this perspective. Condition (cm3) uses the idea that closure in FCA (i.e. the formation of $\alpha_P(\omega_P(X))$ from some X) can be understood as logical consequence, i.e. $X' \subseteq \alpha_P(\omega_P(X))$ means that X carries more knowledge than X', as remarked in Condition (cm3). Thus it allows us to strenghten on the left-hand side of the relation, and to weaken on the right-hand side.

We show now that we indeed obtain a category.

Proposition 1 (identity context morphism). *With notation as in Definition 3, the relation ι_P defined by*

$$X \iota_P Y \text{ iff } Y \subseteq \alpha(\omega(X))$$

defines a context morphism, which we call the identity context morphism.

Proof. Conditions (cm1) and (cm2) are obviously satisfied. Condition (cm3) follows from monotonicity of $\alpha \circ \omega$ and the fact that $\alpha \circ \omega$ is idempotent. □

Composition of context morphisms is composition of relations, so there is nothing to show in this respect.

Theorem 3. *Formal contexts together with context morphisms constitute a category* **Cxt**.

Proof. We first show that the composition of two context morphisms is a context morphism. So let $\rightarrowtriangle_{PQ}$ and $\rightarrowtriangle_{QR}$ be context morphisms. Then condition (cm1) is easily verified for $(\rightarrowtriangle_{QR} \circ \rightarrowtriangle_{PQ})$. Concerning (cm2), assume $X(\rightarrowtriangle_{QR} \circ \rightarrowtriangle_{PQ})Y_1$ and $X(\rightarrowtriangle_{QR} \circ \rightarrowtriangle_{PQ})Y_2$. Then there exist $Z_1, Z_2 \in \mathsf{Fin}(Q_a)$ with $X \rightarrowtriangle_{PQ} Z_1$, $X \rightarrowtriangle_{PQ} Z_2$, hence $X \rightarrowtriangle_{PQ} Z_1 \cup Z_2$, as well as $Z_i \rightarrowtriangle Y_i$ for $i = 1, 2$. Since $Z_i \subseteq \alpha_Q(\omega_Q(Z_1 \cup Z_2))$ and $Y_i \subseteq \alpha_R(\omega_R(Y_i))$ for $i = 1, 2$, we conclude by (cm3) that $Z_1 \cup Z_2 \rightarrowtriangle_{QR} Y_i$ for $i = 1, 2$, and by (cm2) that $Z_1 \cup Z_2 \rightarrowtriangle_{QR} Y_1 \cup Y_2$ which suffices. For (cm3), assume $X' \subseteq \alpha_P(\omega_P(X))$, $X'(\rightarrowtriangle_{QR} \circ \rightarrowtriangle_{PQ})Y'$, and $Y \subseteq \alpha_R(\omega_R(Y'))$. Then there exists $Z \in \mathsf{Fin}(Q_a)$ with $X' \rightarrowtriangle_{PQ} Z$ and $Z \rightarrowtriangle_{QR} Y'$. Since $Z \subseteq \alpha_Q(\omega_Q(Z))$, we conclude by (cm3) that $X \rightarrowtriangle_{PQ} Z$ and $Z \rightarrowtriangle_{QR} Y$, hence $X(\rightarrowtriangle_{QR} \circ \rightarrowtriangle_{PQ})Y$ by definition of composition, as desired.

The remaining conditions are easily verified: associativity of morphisms follows from the fact that composition of morphisms is composition of relations. The identity axiom follows from (cm3). □

5 Information Systems

We show that the category **Cxt** is equivalent to the cartesian closed category of complete algebraic lattices with Scott continuous functions. Our proof utilizes the fact that the latter category is eqivalent to the category of Scott information systems with trivial consistency predicate and approximable mappings as morphisms. The corresponding definitions are as follows, and can be found in [12, 17,18,22,26,28].

An *information system (with trivial consistency predicate)* $\underline{A}$ is a pair $(A, \vdash_A)$, where A is the *token set* and $\vdash_A$ $\mathsf{Fin}(A) \times \mathsf{Fin}(A)$ is the *entailment relation,* and furthermore the conditions

(is1) $a \in X$ implies $X \vdash_A \{a\}$

(is2) $(\forall b \in Y. X \vdash_A \{b\})$ and $Y \vdash_A Z$ imply $X \vdash_A Z$

are satisfied for all $a \in A$ and $X, Y, Z \in \mathsf{Fin}(A)$. An *information state* of an information system is a set $X \subseteq A$ such that $\{a\} \in X$ whenever there is $Y \in \mathsf{Fin}(X)$ with $Y \vdash_A \{a\}$. Information states can be characterized as the images of the operator $\mathsf{state} : 2^A \to 2^A$ defined by

$$\mathsf{state}(X) = \{a \mid \exists Y(Y \in \mathsf{Fin}(X) \text{ and } Y \vdash_A \{a\})\}.$$

The set of all information states of an information system $\underline{A}$ is denoted by $\mathsf{states}(A)$.

Let $\underline{A}$ and $\underline{B}$ be information systems. An *approximable mapping* $\rightsquigarrow_{AB} = \rightsquigarrow$ from $\underline{A}$ to $\underline{B}$ is a relation $\rightsquigarrow \subseteq \mathsf{Fin}(A) \times \mathsf{Fin}(B)$, such that the following conditions are satisfied for all $X, X', Y_1, Y_2 \in \mathsf{Fin}(A)$ and $Y, Y' \in \mathsf{Fin}(B)$.

(am1) $\emptyset \rightsquigarrow \emptyset$
(am2) $X \rightsquigarrow Y_1$ and $X \rightsquigarrow Y_2$ imply $X \rightsquigarrow Y_1 \cup Y_2$
(am3) $X \vdash_A X'$ and $X' \rightsquigarrow Y'$ and $Y' \vdash_B Y$ imply $X \rightsquigarrow Y$

Information systems with trivial consistency predicate together with approximable mappings as morphisms constitute a cartesian closed category, which we denote by **ISys**. The identity morphisms in this case are given by $X \iota_A Y$ iff $X \vdash_A Y$, for any information system $\underline{A}$. Composition of morphisms is composition of relations.

The following definition and theorem are taken from [26].

Definition 4. *For a given formal context $P = (P_o, P_a, \models)$, define a system $\mathcal{IS}(P) = (P_a, \vdash)$ by setting $X \vdash Y$ iff $Y \subseteq \alpha(\omega(X))$.*

Theorem 4. *Given a formal context $P = (P_o, P_a, \models)$, we have that $\mathcal{IS}(P)$ is an information system. Furthermore, a subset $X \subseteq P_a$ is an approximable concept of P if and only if it is a state of the derived information system $\mathcal{IS}(P)$.*

The mapping $\mathcal{IS}$ will later on turn out to be the object part of a functor. The object part of the corresponding left adjoint $\mathcal{CT}$ will be defined next.

Definition 5. *Given an information system $\underline{A} = (A, \vdash)$, let $\mathcal{CT}(A)$ be the formal context $(\mathsf{states}(A), K(\mathsf{states}(A)), \subseteq)$, where $K(\mathsf{states}(A))$ stands for the set of compact elements of $(\mathsf{states}(A), \subseteq)$.*

The following is taken from [26].

Theorem 5. *The approximable concepts of $\mathcal{CT}(A)$ coincide with the downward-closed directed sets of compact elements of the complete algebraic lattice*

$$(\mathsf{states}(A), \subseteq).$$

Hence (by ideal completion), $\mathcal{A}(\mathcal{CT}(A))$ is isomorphic to $(\mathsf{states}(A), \subseteq)$ via the isomorphism $K \mapsto \sup K$, where $\mathcal{A}$ is defined in Theorem 2.

We describe the action of $\mathcal{IS}$ and $\mathcal{CT}$ on morphisms in order to obtain functors between the respective categories.

Let $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$ be formal contexts, and let $\rightarrowtriangle$ be a context morphism. Let $\mathcal{IS}(P) = (P_a, \vdash_P)$ and $\mathcal{IS}(Q) = (Q_a, \vdash_Q)$. Then define $\mathcal{IS}(\rightarrowtriangle_{PQ}) = \rightsquigarrow \subseteq \mathsf{Fin}(P_a) \times \mathsf{Fin}(Q_a)$ by setting $X \rightsquigarrow Y$ iff $X \rightarrowtriangle Y$.

Theorem 6. *The relation $\rightsquigarrow$ is an approximable mapping and $\mathcal{IS}$ is a functor from* **Cxt** *to* **ISys**.

Proof. Straightforward by inspecting the defining properties of a functor. □

Concerning $\mathcal{CT}$, let $\underline{A} = (A, \vdash_A)$ and $\underline{B} = (B, \vdash_B)$ be information systems and let

$$\mathcal{CT}(A) = P = (\mathsf{states}(A), \mathsf{K}(\mathsf{states}(A)), \subseteq) \qquad \text{and}$$
$$\mathcal{CT}(B) = Q = (\mathsf{states}(B), \mathsf{K}(\mathsf{states}(B)), \subseteq)$$

be corresponding formal contexts as defined in Definition 5. Furthermore, let $\rightsquigarrow$ be an approximable mapping from A to B. Then define

$$\mathcal{CT}(\rightsquigarrow) = \rightarrowtriangle \subseteq \mathsf{Fin}(\mathsf{K}(\mathsf{states}(A))) \times \mathsf{Fin}(\mathsf{K}(\mathsf{states}(B)))$$

by setting $X \rightarrowtriangle Y$ iff for each $Y' \in \mathsf{Fin}(\bigcup Y)$ there exists $X' \in \mathsf{Fin}(\bigcup X)$ with $X' \rightsquigarrow Y'$.

Lemma 1. *The relation $\rightarrowtriangle$ is a context morphism.*

Proof. For (cm1) note that $\mathsf{Fin}(\bigcup \emptyset) = \{\emptyset\}$.

For (cm2) let $X \rightarrowtriangle Y_1$ and $X \rightarrowtriangle Y_2$, i.e. for each $Y_1' \in \mathsf{Fin}(\bigcup Y_1)$ there exists $X' \in \mathsf{Fin}(\bigcup X)$ with $X' \rightsquigarrow Y_1'$, and for each $Y_2' \in \mathsf{Fin}(\bigcup Y_2)$ there exists $X'' \in \mathsf{Fin}(\bigcup X)$ with $X'' \rightsquigarrow Y_2'$. Now let $Y \in \mathsf{Fin}(\bigcup(Y_1 \cup Y_2))$. Then there exist $Y_1' \in \mathsf{Fin}(\bigcup Y_1)$ and $Y_2' \in \mathsf{Fin}(\bigcup Y_2)$ with $Y_1' \cup Y_2' = Y$. So we also have $X' \in \mathsf{Fin}(\bigcup X)$ with $X' \rightsquigarrow Y_1'$ and $X'' \in \mathsf{Fin}(\bigcup X)$ with $X'' \rightsquigarrow Y_2'$. Since $X' \cup X'' \in \mathsf{Fin}(\bigcup X)$ and $X' \cup X'' \vdash_A X'$ and $X' \cup X'' \vdash_A X''$, we obtain by (am3) that $X' \cup X'' \rightsquigarrow Y_i'$ (for $i = 1, 2$), and hence $X' \cup X'' \rightsquigarrow Y_1' \cup Y_2' = Y$. By $X' \cup X'' \in \mathsf{Fin}(\bigcup X)$ we conclude $X \rightarrowtriangle Y_1 \cup Y_2$.

For (cm3), note that for $X, Y \in \mathsf{Fin}(\mathsf{K}(\mathsf{state}(A)))$ we have $X \subseteq \alpha_P(\omega_P(Y))$ if and only if $\bigcup X \subseteq \bigcup Y$. Now assume that $X' \subseteq \alpha_P(\omega_P(X))$ and $X' \rightarrowtriangle Y'$ and $Y \subseteq \alpha_Q(\omega_Q(Y'))$. Then $\bigcup X' \subseteq \bigcup X$ and $\bigcup Y \subseteq \bigcup Y'$ and for each $Y'' \in \mathsf{Fin}(\bigcup Y')$ there exists $X'' \in \mathsf{Fin}(\bigcup X')$ with $X'' \rightsquigarrow Y''$. But then in particular, for each $Y'' \in \mathsf{Fin}(\bigcup Y)$ there exists $X'' \in \mathsf{Fin}(\bigcup X)$ with $X'' \rightsquigarrow Y''$. So $X \rightarrowtriangle Y$. □

Theorem 7. *$\mathcal{CT}$ is a functor from* **ISys** *to* **Cxt**.

Proof. Concerning the identity condition, we have $X\,\mathcal{CT}(\mathrm{id}_A)Y$ iff for each $Y' \in \mathsf{Fin}(\bigcup Y)$ there exists $X' \in \mathsf{Fin}(\bigcup X)$ with $X' \vdash_A Y'$. Or in other words, we have $X\,\mathcal{CT}(\mathrm{id}_A)Y$ iff for each $Y' \in \mathsf{Fin}(\bigcup Y)$ with $\mathsf{state}(Y') \subseteq \mathsf{state}(\bigcup Y)$ there exists

$X' \in \mathsf{Fin}(\bigcup X)$ with $\mathsf{state}(X') \subseteq \mathsf{state}(\bigcup X)$ and $\mathsf{state}(Y') \subseteq (X')$. Since finitely generated states are compact in the complete algebraic lattice $(\mathsf{states}(A), \subseteq)$, this is equivalent to the statement $\mathsf{state}(\bigcup Y) \subseteq \mathsf{state}(\bigcup X)$, or in other words, $Y \subseteq \alpha(\omega(X))$.

It remains to show that $\mathcal{CT}(\rightsquigarrow_{BC} \circ \rightsquigarrow_{AB}) = \mathcal{CT}(\rightsquigarrow_{BC}) \circ \mathcal{CT}(\rightsquigarrow_{AB})$. Let first $X\,\mathcal{CT}(\rightsquigarrow_{BC}) \circ \mathcal{CT}(\rightsquigarrow_{AB}) Y$, i.e. there is Z with $X\,\mathcal{CT}(\rightsquigarrow_{BC})Z$ and $Z\,\mathcal{CT}(\rightsquigarrow_{AB})Y$. This means that for all $Y' \in \mathsf{Fin}(\bigcup Y)$ there exists $Z' \in \mathsf{Fin}(\bigcup Z)$ with $Z' \rightsquigarrow_{BC} Y'$ and for all $Z' \in \mathsf{Fin}(\bigcup Z)$ there exists $X' \in \mathsf{Fin}(\bigcup X)$ with $X' \rightsquigarrow_{AB} Z'$. Consequently, for all $Y' \in \mathsf{Fin}(\bigcup Y)$ there is $X' \in \mathsf{Fin}(\bigcup X)$ with $X'(\rightsquigarrow_{BC} \circ \rightsquigarrow_{AB})Y'$, i.e. $X\,\mathcal{CT}(\rightsquigarrow_{BC} \circ \rightsquigarrow_{AB})Y$.

Conversely, let $X\,\mathcal{CT}(\rightsquigarrow_{BC} \circ \rightsquigarrow_{AB})Y$, i.e. for all $Y' \in \mathsf{Fin}(\bigcup Y)$ there exists $X' \in \mathsf{Fin}(\bigcup X)$ with $X'(\rightsquigarrow_{BC} \circ \rightsquigarrow_{AB})Y'$. Hence, for all $Y' \in \mathsf{Fin}(\bigcup Y)$ there exist $X' \in \mathsf{Fin}(\bigcup X)$ and $Z' \in \mathsf{Fin}(Q_a)$ with $X' \rightsquigarrow_{AB} Z'$ and $Z' \rightsquigarrow_{BC} Y'$. Let $Y = \{Y_1, \ldots, Y_n\}$. Then each Y_i is a compact state, hence for each i there is $Y_i' \in \mathsf{Fin}(\bigcup Y)$ with $\mathsf{state}(Y_i') = \mathsf{state}(Y_i)$. For each such Y_i' there exist $Z_i' \in \mathsf{Fin}(Q_a)$ and $X_i' \in \mathsf{Fin}(\bigcup X)$ with $X_i' \rightsquigarrow_{AB} Z_i'$ and $Z_i' \rightsquigarrow_{BC} Y_i'$. Let $Z = \{\mathsf{state}(Z_1'), \ldots, \mathsf{state}(Z_n')\}$. Now, given $Y' \in \mathsf{Fin}(\bigcup Y)$ we obtain $\bigcup Y_i' \vdash_C Y'$. By (am2) and (am3) we have $\bigcup_i Z_i' \rightsquigarrow_{BC} \bigcup Y_i'$, and by (am3) we obtain $\bigcup_i Z_i' \rightsquigarrow_{BC} Y'$. Since $\bigcup_i Z_i' \in \mathsf{Fin}(\bigcup Z)$, we obtain $Z\,\mathcal{CT}(\rightsquigarrow_{BC})Y$. The same kind of argument along $\rightsquigarrow_{AB}$ yields $X\,\mathcal{CT}(\rightsquigarrow_{BC})Z$, which completes the proof. □

6 Categorical Equivalence and Cartesian Closedness

In this section we establish the fact that $\mathcal{IS}$ and $\mathcal{CT}$ provide equivalences between categories.

Recall that a morphism $f \in \mathbf{C}(A, B)$ is called an *isomorphism* if there is a (necessarily unique) morphism $g \in \mathbf{C}(B, A)$ such that $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. The *identity functor* on a category $\mathbf{C}$ maps all objects and morphisms to themselves, and is denoted by $\mathrm{id}_{\mathbf{C}}$. A *natural transformation* $\eta : \mathbf{F} \Rightarrow \mathbf{G}$ between functors $\mathbf{F}, \mathbf{G} : \mathbf{A} \to \mathbf{B}$ is a class of morphisms $\eta_A \in \mathbf{B}(\mathbf{F}A, \mathbf{G}A)_{A \in |\mathbf{A}|}$ such that for every $f \in \mathbf{A}(A, A')$ we have $\eta_{A'} \circ \mathbf{F}f = \mathbf{G}f \circ \eta_A$. A natural transformation is called a *natural isomorphism* if all of its morphisms are isomorphisms. A functor $\mathbf{F}$ from $\mathbf{A}$ to $\mathbf{B}$ is called an *equivalence of categories* iff there is a functor $\mathbf{G}$ from $\mathbf{B}$ to $\mathbf{A}$ and two natural isomorphisms $\mathrm{id}_{\mathbf{B}} \Rightarrow \mathbf{FG}$ and $\mathbf{GF} \Rightarrow \mathrm{id}_{\mathbf{A}}$. $\mathbf{G}$ is then also an equivalence of categories, and is left adjoint to $\mathbf{F}$.

Lemma 2. *There exists a natural transformation $\eta : \mathcal{CT} \circ \mathcal{IS} \Rightarrow \mathrm{id}_{\mathbf{Cxt}}$, i.e. a class of context morphism $(\eta_P)_P$ from $\mathcal{CT}(\mathcal{IS}(P))$ to P, where P ranges over all formal contexts, such that for every context morphism $\rightarrowtriangle$ between formal contexts P and Q we have $\eta_Q \circ \mathcal{CT}(\mathcal{IS}(\rightarrowtriangle)) = \rightarrowtriangle \circ \eta_P$. Furthermore, η is a natural isomorphism, i.e. all η_P are isomorphisms – in other words, for each η_P there exists a context morphism $\rightarrowtriangle_P$ from P to $\mathcal{CT}(\mathcal{IS}(P))$ such that $\rightarrowtriangle_P \circ \eta_P = \iota_P$ and $\eta_P \circ \rightarrowtriangle_P = \iota_{\mathcal{CT}(\mathcal{IS}(P))}$.*

Proof. Let $P = (P_o, P_a, \models_P)$ be some formal context. We define η_P by setting $S\eta_P X$,for $S \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(\mathcal{IS}(P))))$ and $X \in \mathsf{Fin}(P_a)$, whenever $X \in \mathsf{Fin}(\bigcup S)$. It is easily verified that η_P is a context morphism.

Now let $T(\eta_Q \circ \mathcal{CT}(\mathcal{IS}(\rightarrowtriangle)))X$. This is equivalent to saying that there exists some $S \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(\mathcal{IS}(P))))$ such that $X \in \mathsf{Fin}(\bigcup S)$ and for all $s \in \mathsf{Fin}(\bigcup S)$ there exists some $t \in \mathsf{Fin}(\bigcup T)$ with $t \rightarrowtriangle s$. This implies that there is $t \in \mathsf{Fin}(\bigcup T)$ with $t \rightarrowtriangle X$, which in turn is equivalent to $T(\rightarrowtriangle \circ \eta_P)X$. Conversely, let $T(\rightarrowtriangle \circ \eta_P)X$, which is equivalent to saying that there is $t \in \mathsf{Fin}(\bigcup T)$ with $t \rightarrowtriangle X$. Now with $S = \{\mathsf{state}(X)\} \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(\mathcal{IS}(P))))$ this implies that $X \in \mathsf{Fin}(\bigcup S)$ and (by (cm3)) for all $s \in \mathsf{Fin}(\bigcup S)$ there exists $t \in \mathsf{Fin}(\bigcup T)$ with $t \rightarrowtriangle s$. This is in turn equivalent to $T(\eta_Q \circ \mathcal{CT}(\mathcal{IS}(\rightarrowtriangle)))X$, as noted earlier in this paragraph.

To show that all η_P are isomorphisms, let $\rightarrowtriangle_P$ be the context morphism from P to $\mathcal{CT}(\mathcal{IS}(P))$ which is defined by $X \rightarrowtriangle_P S$, for $X \in \mathsf{Fin}(P_a)$ and $S \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(\mathcal{IS}(P))))$, whenever $\bigcup S \subseteq \alpha_P(\omega_P(X))$. It is easily verified that $\rightarrowtriangle_P$ is indeed a context morphism. Now $X(\eta_P \circ \rightarrowtriangle_P)Y$ iff there exists S with $X \rightarrowtriangle_P S$ and $S\eta_P Y$, i.e. iff there exists S with $\bigcup S \subseteq \alpha_P(\omega_P(X))$ and $Y \in \mathsf{Fin}(\bigcup S)$. This in turn is equivalent to $Y \in \mathsf{Fin}(\alpha_P(\omega_P(X)))$, i.e. to $X\iota_P Y$ as desired. Likewise, let $T, S \in \mathsf{Fin}(\mathsf{K}(\mathsf{state}(\mathcal{IS}(P))))$. Then $S(\rightarrowtriangle_P \circ \eta_P)T$ iff there exists Y with $S\eta_P Y$ and $Y \rightarrowtriangle_P T$, i.e.iff $Y \in \mathsf{Fin}(\bigcup S)$ and $\bigcup T \subseteq \alpha_P(\omega_P(Y))$. Since S and T are sets of compact states, this in turn is equivalent to $\bigcup T \subseteq \bigcup S$, i.e. to $T\iota_{\mathcal{CT}(\mathcal{IS}(P))}S$. □

Lemma 3. *There exists a natural transformation* $\eta : \mathrm{id}_{\mathbf{ISys}} \Rightarrow \mathcal{IS} \circ \mathcal{CT}$*, i.e. a class of approximable mappings* $(\eta_A)_{\underline{A}}$ *from* $\underline{A}$ *to* $\mathcal{IS}(\mathcal{CT}(A))$*, where* $\underline{A}$ *ranges over all information systems (with trivial consistency predicate), such that for every approximable mapping* $\leadsto$ *between information systems* $\underline{A}$ *and* $\underline{B}$ *we have* $\eta_B \circ \leadsto = \mathcal{IS}(\mathcal{CT}(\leadsto)) \circ \eta_A$*. Furthermore,* η *is a natural isomorphism, i.e. all* η_A *are isomorphisms — or in other words, for each* η_A *there exists a context morphism* $\leadsto_A$ *from* $\mathcal{IS}(\mathcal{CT}(A)))$ *to* $\underline{A}$ *such that* $\leadsto_A \circ \eta_A = \iota_A$ *and* $\eta_A \circ \leadsto_A = \iota_{\mathcal{IS}(\mathcal{CT}(A))}$.

Proof. Let $\underline{A} = (A, \vdash)$ be some information system. We define η_A by setting $X\eta_A S$, for $X \in \mathsf{Fin}(A)$ and $S \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(A)))$, whenever $X \vdash_A s$ for all $s \in \mathsf{Fin}(\bigcup S)$. It is easily verified that η_A is an approximable mapping.

We have $X(\eta_B \circ \leadsto)T$ iff there exists $Y \in \mathsf{Fin}(B)$ such that $X \leadsto Y$ and for all $t \in \mathsf{Fin}(\bigcup T)$ we have $Y \vdash_B t$. This is equivalent to the statement that

(♯) for all $t \in \mathsf{Fin}(\bigcup T)$ we have $X \leadsto t$.

On the other hand, we have $X(\mathcal{IS}(\mathcal{CT}(\leadsto)) \circ \eta_A)T$ if and only if

($*$) there exists $S \in \mathsf{Fin}(\mathsf{K}(\mathsf{states}(A)))$ such that for all $s \in \mathsf{Fin}(\bigcup S)$ we have $X \vdash_A s$ and for all $t \in \mathsf{Fin}(\bigcup T)$ there exists $u \in \mathsf{Fin}(\bigcup S)$ with $u \leadsto t$.

Statement (#) implies ($*$) via $S = \{\mathsf{state}(X)\}$. Statement ($*$) implies that for all $t \in \mathsf{Fin}(\bigcup T)$ there exists $u \in \mathsf{Fin}(\bigcup S)$ with $X \vdash_A u$ and $u \leadsto t$, which by (am3) implies (#).

To show that all η_A are isomorphisms, let $\rightsquigarrow_A$ be the approximable mapping from $\mathcal{IS}(\mathcal{CT}(A))$ to $\underline{A}$ defined by $S \rightsquigarrow_A X$, for all $S \in \mathsf{Fin}(\mathsf{K}(\mathbf{states}(A)))$ and $X \in \mathsf{Fin}(A)$, whenever $X \in \mathsf{Fin}(\bigcup S)$. It is easily verified that $\rightsquigarrow_A$ is an approximable mapping. Now $X(\rightsquigarrow_A \circ \eta_A)Y$ if and only if there exists $S \in \mathsf{Fin}(\mathsf{K}(\mathbf{states}(A)))$ with $X\eta_A S$ and $S \rightsquigarrow_A Y$, i.e. $X \vdash_A s$ for all $s \in \mathsf{Fin}(\bigcup S)$ and $Y \in \mathsf{Fin}(\bigcup S)$. This in turn is equivalent to $X \vdash_A Y$, i.e. to $X \iota_A Y$ as desired. Likewise, $S(\eta_A \circ \rightsquigarrow_A)T$ if and only if there exists $X \in \mathsf{Fin}(A)$ with $S \rightsquigarrow_A X$ and $X\eta_A T$. This in turn is equivalent to saying that there exists $X \in \mathsf{Fin}(\bigcup S)$ such that $X \vdash_A t$ for all $t \in \mathsf{Fin}(\bigcup T)$. Since S and T are sets of compact states, this is equivalent to the statement $\bigcup T \subseteq \bigcup S$, i.e. to $S\iota_{\mathcal{IS}(\mathcal{CT}(A))}T$. □

Theorem 8. *Both $\mathcal{CT}$ and $\mathcal{IS}$ are equivalences of categories, i.e. the categories* **Cxt** *and* **ISys** *are equivalent, and* **Cxt** *is cartesian closed.*

Proof. It suffices to show that there are natural isomorphisms $\mathcal{CT} \circ \mathcal{IS} \Rightarrow \mathrm{id}_{\mathbf{Cxt}}$ and $\mathrm{id}_{\mathbf{ISys}} \Rightarrow \mathcal{IS} \circ \mathcal{CT}$, which were provided by Lemmata 2 and 3. The last statement follows from the well-known fact that **ISys** is cartesian closed. □

7 Constructions

We have established our main result in Theorem 8 and shown that **Cxt** is cartesian closed. We now study what the corresponding categorical constructions, i.e. product and function space, look like. Although the existence of these constructions has been justified in previous sections, carrying out the actual constructions in full detail can be an engineering endeavor [22,28].

Terminal Object

The *unit* or *terminal object* in **Cxt** is the context $\mathbf{1} = (\emptyset, \emptyset, \emptyset)$, and for each context $P = (P_o, P_a, \models)$ the unique context morphism $\rightarrowtriangle$ from P to $\mathbf{1}$ is given by $X \rightarrowtriangle \emptyset$ for all $X \in \mathsf{Fin}(P_a)$.

Product

For a formal context $P = (P_o, P_a, \models_P)$ let $P' = (P'_o, P'_a, \models_{P'})$, where $P'_o = P_o \cup \{g\}$, $P'_a = P_a \cup \{m\}$, and g, m are some elements not in P_o respectively P_a. $\models_{P'}$ coincides with $\models_P$ on $P_a \times P_o$ and $g \models_{P'} a$ for all $a \in P_a \cup \{m\}$ and $h \models_{P'} m$ for all $h \in P_o \cup \{g\}$. Informally, P' is obtained from P by "adding a full row and a full column".

Let $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$ be formal contexts. Define the *product*

$$P \times Q = (P'_o \times Q'_o, P'_a \times Q'_a, \models_{P \times Q})$$

of P and Q by setting $(g_1, g_2) \models_{P \times Q} (m_1, m_2)$ iff $g_1 \models_{P'} m_1$ and $g_2 \models_{Q'} m_2$. Obviously, $P \times Q$ is a formal context.

Theorem 9. *Let $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$ be formal contexts. Then there exist context morphisms $\pi_P : P \times Q \to P$ and $\pi_Q : P \times Q \to Q$ such that for all context morphisms $\rightarrowtriangle_P$ from R to P and $\rightarrowtriangle_Q$ from R to Q there exists exactly one context morphism $\langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle$ from R to $P \times Q$ such that*

$$\pi_P \circ \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle = \rightarrowtriangle_P \qquad \textit{and} \qquad \pi_Q \circ \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle = \rightarrowtriangle_Q .$$

Proof. We define π_P by setting $\{(m_i, m_i') \mid i = 1, \ldots, k\} \pi_P \{n_j \mid j = 1, \ldots, l\}$ iff $\{n_j \mid j = 1, \ldots, l\} \subseteq \alpha_P(\omega_P(\{m_i \mid i = 1, \ldots, k\}))$. This is equivalent to saying that $M \pi_P N$ iff $(\pi_1(M) \cap P_a) \iota_P N$, where $\pi_1(M)$ denotes the projection of the set of pairs M to the set of the first components of its elements. The context morphism π_Q is defined analogously. Define $X \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle Y$ iff $X \rightarrowtriangle_P (\pi_1(Y) \cap P_a)$ and $X \rightarrowtriangle_Q (\pi_2(Y) \cap Q_a)$, where π_2 is the corresponding projection on the second component.

We next show $\pi_P \circ \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle = \rightarrowtriangle_P$. Denoting $\langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle$ by $\rightsquigarrow$, we then obtain $X(\pi_P \circ \rightsquigarrow)Y$ iff there exists some $Z \in \mathsf{Fin}(P_a' \times Q_a')$ with $X \rightsquigarrow Z$ and $Z \pi_P Y$, which is the case iff $(*)$ there exists Z with $X \rightarrowtriangle_P (\pi_1(Z) \cap P_a)$, $X \rightarrowtriangle_Q (\pi_2(Z) \cap Q_a)$ and $(\pi_1(Z) \cap P_a) \iota_P Y$. This in turn implies that there is $Z \in \mathsf{Fin}(P_a' \times Q_a')$ with $X \rightarrowtriangle_P Y$ and $X \rightarrowtriangle_Q (\pi_2(Z) \cap Q_a)$. Such a Z trivially exists: every Z with $\pi_2(Z) = \emptyset$ satisfies the condition. So this condition reduces to $X \rightarrowtriangle_P Y$ as required. Conversely, assume $X \rightarrowtriangle_P Y$. Let $Z = Y \times (Q_a' \setminus Q_a)$. Then this implies condition $(*)$ which was shown above to be equivalent to $X(\pi_P \circ \rightsquigarrow)Y$. The equation $\pi_Q \circ \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle = \rightarrowtriangle_Q$ is shown similarly.

For uniqueness assume that there is $\rightarrowtriangle : R \to P \times Q$ which satisfies $\pi_P \circ \rightarrowtriangle = \rightarrowtriangle_P$ and $\pi_Q \circ \rightarrowtriangle = \rightarrowtriangle_Q$. Then we obtain

$$\begin{array}{lllll}
X \rightarrowtriangle Z & \text{iff} & X \rightarrowtriangle Z, & (\pi_1(Z) \cap P_a)\, \iota_P\, (\pi_1(Z) \cap P_a), & \\
 & & & \text{and} \quad (\pi_2(Z) \cap Q_a)\, \iota_Q\, (\pi_2(Z) \cap Q_a) & \\
 & \text{iff} & X \rightarrowtriangle Z, & Z\, \pi_P\, (\pi_1(Z) \cap P_a), \quad \text{and} & Z\, \pi_Q\, (\pi_2(Z) \cap Q_a) \\
 & \text{iff} & X(\pi_P \circ \rightarrowtriangle)(\pi_1(Z) \cap P_a) & \text{and} & X(\pi_Q \circ \rightarrowtriangle)(\pi_2(Z) \cap Q_a) \\
 & \text{iff} & X \rightarrowtriangle_P (\pi_1(Z) \cap P_a) & \text{and} \quad X \rightarrowtriangle_Q (\pi_2(Z) \cap Q_a) &
\end{array}$$

which shows $\rightarrowtriangle = \langle \rightarrowtriangle_P, \rightarrowtriangle_Q \rangle$. □

Function Space

Given formal contexts $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$, we define the function space $P \to Q$ as follows. Consider the set $M = \mathsf{Fin}(P_a) \times \mathsf{Fin}(Q_a)$. We define a Scott information system on M. For all collections $\{(u_i, u_i')\} \in \mathsf{Fin}(M)$ and $\{(v_j', v_j')\} \in \mathsf{Fin}(M)$, let $\{(u_i, u_i')\} \vdash_M \{(v_j, v_j')\}$ iff

$$v_j' \subseteq \alpha_Q \left(\omega_Q \left(\bigcup \{ u_i' \mid u_i \subseteq \alpha_P(\omega_P(v_j)) \} \right) \right)$$

for all j. It is easily verified that $(M, \vdash_M)$ is a Scott information system: In order to show condition (is2), assume that we have $\{(y_i, y_i')\} \vdash_M \{(z, z')\}$ and

$\{(x_j,x'_j)\} \vdash_M (y_i,y'_i)$ for all i. Then $z' \subseteq \alpha_Q\left(\omega_Q\left(\bigcup\{y'_i \mid y_i \subseteq \alpha_P(\omega_P(z))\}\right)\right)$ and $y'_i \subseteq \alpha_Q\left(\omega_Q\left(\bigcup\{x'_j \mid x_j \subseteq \alpha_P(\omega_P(y_i))\}\right)\right)$ for all i. So

$$\begin{aligned}
z' &\subseteq \alpha_Q\left(\omega_Q\left(\bigcup_i \left\{\alpha_Q\left(\omega_Q\left(\bigcup_j \{x'_j \mid x_j \subseteq \alpha_P(\omega_P(y_i))\}\right)\right) \mid y_i \subseteq \alpha_P(\omega_P(z))\right\}\right)\right)\\
&\subseteq \alpha_Q\left(\omega_Q\left(\bigcup_i \left\{\alpha_Q\left(\omega_Q\left(\bigcup_j \{x'_j \mid x_j \subseteq \alpha_P(\omega_P(z))\}\right)\right) \mid y_i \subseteq \alpha_P(\omega_P(z))\right\}\right)\right)\\
&\subseteq \alpha_Q\left(\omega_Q\left(\alpha_Q\left(\omega_Q\left(\bigcup_j \{x'_j \mid x_j \subseteq \alpha_P(\omega_P(z))\}\right)\right)\right)\right)\\
&= \alpha_Q\left(\omega_Q\left(\bigcup \{x'_j \mid x_j \subseteq \alpha_P(\omega_P(z))\}\right)\right)
\end{aligned}$$

as required. The function space $P \to Q$ is then defined as the context

$$\mathcal{CT}(M) = (\mathsf{states}(M), \mathsf{K}(\mathsf{states}(M)), \subseteq).$$

Theorem 10. *Let $P = (P_o, P_a, \models_P)$ and $Q = (Q_o, Q_a, \models_Q)$ be formal contexts. Then the context morphisms from P to Q are exactly the approximable concepts in $P \to Q$.*

Proof. Let A be an approximable concept in $P \to Q$. Then $A \in \mathsf{states}(M)$ by Theorem 5, and we have to show that A is a context morphism. Conditions (cm1) and (cm2) are easily verified. Condition (cm3) follows from the monotonicity of $\alpha_P \circ \omega_P$ and $\alpha_Q \circ \omega_Q$.

Conversely, let A be a context morphism from P to Q. By Theorem 5, we have to show that $A \in \mathsf{states}(M)$. So let $\{(x_i,x'_i)\} \in \mathsf{Fin}(A)$ and $\{(x_i,x'_i)\} \vdash_M \{(a,a')\}$. We have to show that $(a,a') \in A$. From $\{(x_i,x'_i)\} \vdash_M \{(a,a')\}$ we obtain

$$a' \subseteq \alpha_Q\left(\omega_Q\left(\bigcup\{x'_i \mid x_i \subseteq \alpha_P(\omega_P(a))\}\right)\right).$$

Setting $X = \bigcup\{x_i \mid x_i \subseteq \alpha_P(\omega_P(a))\}$ and $X' = \bigcup\{x'_i \mid x_i \subseteq \alpha_P(\omega_P(a))\}$, we obtain $a' \subseteq \alpha_Q(\omega_Q(X'))$ and $X \subseteq \alpha_P(\omega_P(a))$. By (cm2) and (cm3) we also have $(X,X') \in A$, and so by (cm3) again we have $(a,a') \in A$. □

8 Concluding Remarks

We have proposed a notion of morphism for formal contexts which results in a cartesian closed category. Our work is in the spirit of both formal concept analysis and domain theory, and makes way for the cross-transfer of methods and results between these areas. Our work builds on a domain-theoretic perspective on formal concepts, which results in complete algebraic lattices instead of complete lattices as the corresponding concept hierarchies.

Technically, our contribution uses Scott information systems in order to capture the logical content of formal contexts. The general connection between information systems and formal contexts was explicitly outlined in [25]. Such a connection breaks down with classical formal concepts for a certain "discontinuous" class of infinite contexts. This lead to the work reported in [26] in which the notion of approximable concept was introduced and the connection between information systems and formal contexts was established in full generality with the corresponding structure being the complete algebraic lattices. Work along a similar line for the finite case was spelled out in [8] in a different guise using the logic RZ due to [16]. The latter work has also led to the proposal of a non-monotonic reasoning paradigm on (possibly infinite) formal contexts [9], which is in the spirit of the recent evolving answer set programming paradigm [6,20]. Also worth noting are potential connections between our work and that of Lamarche [11] and Plotkin [14].

The work reported here can be viewed as part of a unique research program [8,25,26,27,19] exploiting the synergies among some recurring themes in several independently developing and yet somewhat related areas, such as databases, data-mining, domain theory, logic, and formal concept analysis. Additional interesting connections could be profitably explored with ontological engineering and semantic web. For example, in [27], formal concept analysis has been applied as a formal method for automated web-menu design, where the top layers of the concept lattice naturally provide a menu hierarchy for the navigation of a website. In ontological engineering (e.g. [7,21]), although lattices have been proposed as mathematical structures for representing ontology, FCA provides a scientific and algorithmic basis for it, as well as an understanding that lattice structures are both necessary and sufficient for expressing ontological hierarchies.

Acknowledgment. The first author thanks Markus Krötzsch for some first aid in category theory. We thank the anonymous referees for valuable feedback.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer, Berlin (1999).
2. Barr, M.: *-Autonomous categories and linear logic. *Mathematical Structures in Computer Science* 1 (2), 1991.
3. Barwise, J., Seligman, J.: *Information Flow: the logic of distributed systems.* Cambridge University Press, 1997.
4. Birkhoff, G.: *Lattice Theory.* AMS Colloquium Publications, Vol. 25, 1940.
5. Borceux, F.: Handbook of Categorical Algebra 1: Basic Category Theory. Volume 53 of Encyclopedia of Mathematics and its Applictions. Cambridge University Press (1994).
6. Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F.: A deductive system for nonmonotonic reasoning. In Dix, J., Furbach, U., Nerode, A., eds.: Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97. Volume 1265 of Lecture Notes in Artificial Intelligence., Springer, Berlin (1997).
7. Guarino, N., Poli, R. (eds.): *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers, 1993.
8. Hitzler, P., Wendt, M.: Formal concept analysis and resolution in algebraic domains. In de Moor, A., Ganter, B., eds.: Using Conceptual Structures — Contributions to ICCS 2003, Shaker Verlag, Aachen (2003) 157–170.

9. Hitzler, P.: Default reasoning over domains and concept hierarchies. Technical Report WV–03–14, Knowledge Representation and Reasoning Group, Department of Computer Science, Dresden University of Technology (2003).
10. Johnstone, P.: *Stone Spaces,* Cambridge University Press, 1982.
11. Lamarche, F.: From Chu spaces to cpos. *Theory and Formal Methods of Computing 94,* Imperial College Press, 283–305, 1994.
12. Larsen, K., Winskel, G.: Using information systems to solve recursive domain equations. *Information and Computation,* Vol. 91(2):232–258, 1991.
13. MacLane, S.: *Categories for the Working Mathematician,* Springer-Verlag (1971).
14. Plotkin, G.: Notes on the Chu construction and recursion, manuscript, 1993.
15. Pratt, V.: Chu spaces from the representational viewpoint. *Ann. Pure Appl. Logic* 96 no. 1-3, 319–333, 1999.
16. Rounds, W.C., Zhang, G.-Q.: Clausal logic and logic programming in algebraic domains. Information and Computation **171** (2001) 156–182.
17. Saraswat, V.: The category of constraint systems is Cartesian-closed. In: Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California, June 1992, IEEE Computer Society Press (1992) 341–345.
18. Scott, D.S.: Domains for denotational semantics. In Nielsen, M., Schmidt, E.M., eds.: Automata, Languages and Programming, 9th Colloquium, July 1982, Aarhus, Denmark, Proceedings. Volume 140 of Lecture Notes in Computer Science., Springer, Berlin (1982) 577–613.
19. Shen, G., Ye, T., Sun, J. and Zhang, G.-Q.: Concept lattices, clustering, and visualization using LaTeX, manuscript, 2004.
20. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence (200x), To appear.
21. Sowa, J.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations,* Brooks Cole Publishing Co., 2000.
22. Spreen, D.: A note on strongly finite sequent structures. In Zhang, Lawson, Liu, Luo (eds.), *Domain Theory, Logic and Computation* (Semantic Structures in Computation, Vol. 3), Kluwer Academic Publishers, Dordrecht, 2004.
23. Vickers, S.: *Topology via Logic,* Cambridge University Press, 1989.
24. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered sets,* Reidel, Dordrecht-Boston(1982), 445–470.
25. Zhang, G.-Q.: Chu spaces, formal concepts, and domains. Proceedings of the 19th Conference on the Mathematical Foundations of Programming Semantics, Montreal, Canada, March 19-23, 2003, Electronic Notes in Computer Science Vol. 83, 16 pages, 2003.
26. Zhang, G.-Q., Shen, G.: Approximable concepts, Chu spaces, and information systems. In de Paiva and Pratt (Guest Editors), *Special Issue on Chu Spaces and Applications, Theory and Applications of Categories,* accepted.
27. Zhang, G.-Q., Shen, G., Staiger, J., Troy, A., Sun, J.: *FcAWN* – concept analysis as a formal method for automated web-menu design. Submitted.
28. Zhang, G.-Q.: *Logic of Domains,* Birkhauser, Boston, 1991.

Implications in Triadic Formal Contexts

Bernhard Ganter and Sergei Obiedkov

Technische Universität Dresden, Germany

Abstract. There are different possibilities to define *implications* in a triadic formal context. We discuss several such notions. Our main interest is to give compact descriptions and to use them for an algorithm that generates these implications. Our findings are illustrated by a small example.

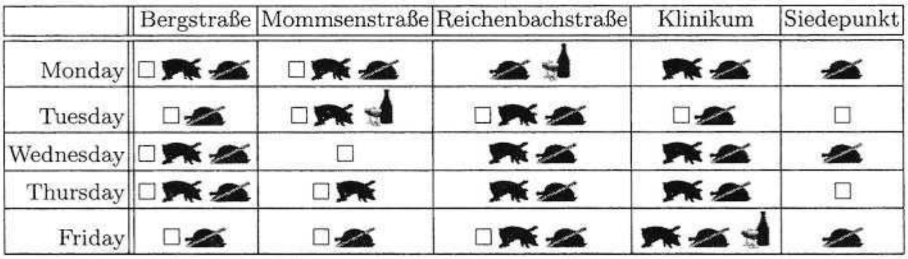
1 Introduction

Triadic concept analysis was introduced by Lehmann and Wille [5] as a natural extension of (dyadic) Formal Concept Analysis.

A **triadic context** $\mathbb{K} := (G, M, B, Y)$ consists of sets G, M, and B (called *objects, attributes,* and *conditions,* respectively), and a ternary relation $Y \subseteq G \times M \times B$. An incidence $(g, m, b) \in Y$ is read as

the object g has the attribute m under the condition b.

The data in Figure 1 may be interpreted as a triadic context. It shows the 5-day meal plan for the five student cafeterias in Dresden[1]. The symbols indicate that meals are served which are vegetarian (🍗), contain meat, but no pork (🐖), contain alcohol (🍷), or have none of theses properties (□). A possible triadic interpretation is obtained by choosing the cafeterias as attributes, the days as objects, and the symbols as conditions. Other assignments will be discussed below.



	Bergstraße	Mommsenstraße	Reichenbachstraße	Klinikum	Siedepunkt
Monday	□ 🐖 🍗	□ 🐖 🍗	🍗 🍷	🐖 🍗	🍗
Tuesday	□ 🍗	□ 🐖 🍷	□ 🐖 🍗	□ 🍗	□
Wednesday	□ 🐖 🍗	□	🐖 🍗	🐖 🍗	🍗
Thursday	□ 🐖 🍗	□ 🐖	🐖 🍗	🐖 🍗	□
Friday	□ 🍗	□ 🍗	□ 🐖 🍗	🐖 🍗 🍷	🍗

Fig. 1. Meal plan for student cafeterias in Dresden.

[1] Jan. 19–23, 2004.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 186–195, 2004.

As an example of how to read the data in Figure 1, consider Thursday at Mommsenstraße: Several meals are served there, at least one of them containing meat, but no pork (🐄), and at least one with pork, but no alcohol (□). No vegetarian meal and no alcoholic meals is offered in Mommsenstraße on Thursday.

Lehmann and Wille in their abovementioned paper have shown how to generalize Formal Concept Analysis to the triadic setting. In particular, they have introduced **triadic formal concepts**. These form a *concept trilattice.* In a subsequent paper, Wille [7] has proved the Basic Theorem of Triadic Concept Analysis, thereby laying the fundament for a systematic mathematical treatment of triadic contexts.

Only a few publications have since then studied this mathematical theory. One of them is the dissertation of Biedermann [1], in which also the study of *triadic implications* is suggested. It is not clear from the beginning what such an implication should be. Biedermann's choice is what he calls *triadic implications*

$$(R \to S)_C,$$

which are to be interpreted as

> *If an object has all attributes from R under all conditions from C, then it also has all attributes from S under all conditions from C.*

We start by considering a class of implications that are stronger. We study expressions of the form

$$R \overset{C}{\to} S, \qquad \text{where } R, S \subseteq M,\ C \subseteq B.$$

We call such an expression a **conditional attribute implication** and read it as "R implies S under all conditions from C ". A conditional attribute implication $R \overset{C}{\to} S$ **holds** in a triadic context $\mathbb{K}$ iff the following is satisfied:

> *For each condition $c \in C$ it holds that if an object $g \in G$ has all the attributes in R then it also has all the attributes in S.*

It is easy to see how the two notions are interrelated:

Proposition 1. $R \overset{C}{\to} S \iff (R \to S)_N$ *for all* $N \subseteq C$.

Our definition gives a system of implications for every subset $C \subseteq B$ of conditions. So in principle we get an exponential family of implication systems, which is unpleasant at least from the standpoint of algorithmic complexity. Even giving a stem base [3] for each such $C \subseteq B$ would not really ease the problem. Therefore, a more systematic approach is needed.

2 The Relative Stem Base

An **attribute implication** $R \to S$ **holds** in a formal context $\mathbb{K} := (G, M, I)$ if R and S are subsets of M and each object $g \in G$ that has all attributes from R

also has all attributes from S. The (usually huge) set of all implications that hold in $\mathbb{K}$ has a canonical irredundant representation (Guigues & Duquenne [4], see also [3]) which we call the **stem base** of $\mathbb{K}$. Is has been observed by Stumme [6] that the stem base can be generalized to the case of background implications. Since we shall make extensive use of this idea, we give a sketch of the method.

Let $\mathbb{K} := (G, M, I)$ be a (dyadic) formal context and let $\mathcal{L}_0$ be a set of implications ("background implications") on M all of which hold in $\mathbb{K}$. We recursively define a **pseudo-intent** of $\mathbb{K}$ **relative to** $\mathcal{L}_0$ (or **relative pseudo-intent** for short) to be a set $P \subseteq M$ with the following properties:

1. P respects $\mathcal{L}_0$
   (i.e., for each $R \to S \in \mathcal{L}_0$ we have that if $R \subseteq P$ then also $S \subseteq P$),
2. $P \neq P''$, and
3. if $Q \subseteq P$, $Q \neq P$, is a relative pseudo-intent of $\mathbb{K}$, then $Q'' \subseteq P$.

The set

$$\mathcal{L}_{\mathbb{K},\mathcal{L}_0} := \{P \to P'' \mid P \text{ relative pseudo-intent of } \mathbb{K}\}$$

is called the **stem base of** $\mathbb{K}$ **relative to** $\mathcal{L}$ (or **relative stem base** for short). It is obvious that all implications in $\mathcal{L}_{\mathbb{K},\mathcal{L}_0}$ hold in $\mathbb{K}$.

Theorem 1 ([4], [6]). *If all implications from $\mathcal{L}_0$ hold in $\mathbb{K}$, then*

1. *each implication that holds in $\mathbb{K}$ follows from $\mathcal{L}_0 \cup \mathcal{L}_{\mathbb{K},\mathcal{L}_0}$, and*
2. *$\mathcal{L}_{\mathbb{K},\mathcal{L}_0}$ is irredundant w.r.t. 1.*

Proof. Let $P \subseteq M$ be a set which respects all implications in $\mathcal{L}_0 \cup \mathcal{L}_{\mathbb{K},\mathcal{L}_0}$. Then P obviously fulfills conditions 1 and 3 of the definition of a relative pseudo-intent. Thus P cannot fulfill condition 2 as well, because it then would be a relative pseudo-intent and the implication $P \to P''$ would be in $\mathcal{L}_{\mathbb{K},\mathcal{L}_0}$, but would not be respected by P. From this it is straightforward to obtain 1.

To see that 2 also holds, note that every relative pseudo-intent P respects all implications in $\mathcal{L}_0 \cup \mathcal{L}_{\mathbb{K},\mathcal{L}_0}$, except for $P \to P''$. Therefore $P \to P''$ does not follow from $\mathcal{L}_0 \cup (\mathcal{L}_{\mathbb{K},\mathcal{L}_0} \setminus \{P \to P''\})$.

3 The Context of Conditional Implications

From a given triadic context $\mathbb{K}$ we construct a dyadic context $\mathbb{C}_{\text{imp}}(\mathbb{K})$ as follows. We use the set

$$\text{Imp}(M) := \{R \to S \mid R, S \subseteq M\}$$

of all possible implications on M as the object set[2] of the dyadic formal context

$$\mathbb{C}_{\text{imp}}(\mathbb{K}) := (\text{Imp}(M), B, I),$$

where

$$(R \to S)\, I\, c \quad :\Longleftrightarrow \quad R \xrightarrow{c} S \text{ holds in } \mathbb{K}.$$

[2] In practice it suffices to use only implications of the form $R \to s$, where $s \in M \setminus R$.

The formal concepts of this context are the pairs $(\mathcal{F}, D)$, for which

- $\mathcal{F}$ is a set of implications,
- D is a set of conditions, such that
- $\mathcal{F}$ is the set of all implications $R \to S$ such that $R \overset{D}{\to} S$ holds, and D is the largest set of conditions for which this is the case.

The concepts of $\mathbb{C}_{\mathbf{imp}}(\mathbb{K})$ structure the set of conditional implications in a nice manner. We shall show later that each *extent* is an *implicational theory,* i.e., the set of all implications of some formal context. Therefore, each extent has a stem base. However, since each extent is the intersection of meet-irreducible extents, it suffices to have these to have full information. The *intents* are condition sets. An *implication* $D \to E$ between such condition sets expresses that if a conditional implication $R \overset{D}{\to} S$ holds then $R \overset{E}{\to} S$ must also hold. So what we get is a lattice of implicational theories hierarchically ordered by the conditions under which they hold.

The diagram in Figure 2 shows the concept lattice of the conditional implications for the data from Figure 1. Not all of the possible implications are used as labels (their total number is 80). Instead, we have given sufficiently many implications to generate the respective implication set in each case. How these are chosen will be discussed below.

For example, the implication

$$\text{Siedepunkt} \to \text{Bergstraße}$$

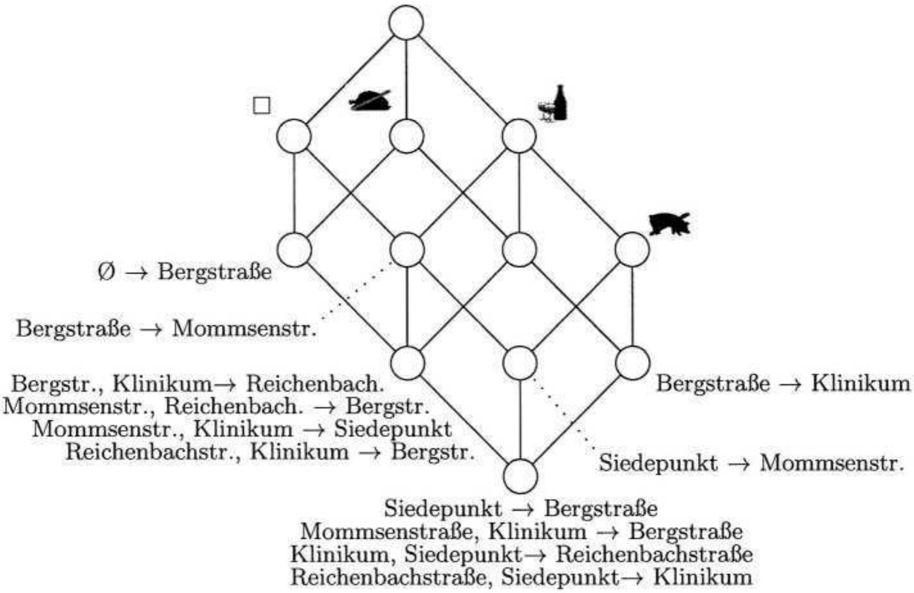

□
∅ → Bergstraße
Bergstraße → Mommsenstr.
Bergstr., Klinikum→ Reichenbach.
Mommsenstr., Reichenbach. → Bergstr.
Mommsenstr., Klinikum → Siedepunkt
Reichenbachstr., Klinikum → Bergstr.
Bergstraße → Klinikum
Siedepunkt → Mommsenstr.
Siedepunkt → Bergstraße
Mommsenstraße, Klinikum → Bergstraße
Klinikum, Siedepunkt→ Reichenbachstraße
Reichenbachstraße, Siedepunkt→ Klinikum

Fig. 2. The lattice of conditional attribute implications.

at the bottom of the diagram expresses that whatever kind of meal is available at the Siedepunkt cafeteria can also be obtained at Bergstraße. The implication

$$\text{Bergstraße} \to \text{Klinikum}$$

on the right hand side says that whenever a vegetarian meal, a meal with alcohol or a meal without pork is served at Bergstraße, the same holds true for Klinikum. And the implication

$$\emptyset \to \text{Bergstraße}$$

on the left comes from the fact that the Bergstraße cafeteria serves a vegetarian meal and a □-meal every day.

Our labelling of the diagram is different from the usual one and requires some explanation. The attributes of $\mathbb{C}_{\text{imp}}(\mathbb{K})$ are the conditions of the triadic context $\mathbb{K}$ and are used as labels in the usual manner. So in particular each meet-irreducible concept has (at least) one condition label. The extent of this concept is the set of all implications that hold under this condition in $\mathbb{K}$. Each other extent is the intersection of such.

The object labelling is more tricky. The objects of $\mathbb{C}_{\text{imp}}(\mathbb{K})$ are implications, and thus the object labelling requires to write implications to lattice elements. There are two standard ways to do this, both of which are not fully satisfactory in our case. One is to write each object of $\mathbb{C}_{\text{imp}}(\mathbb{K})$ to the corresponding object concept. This would require to label the diagram with 80 implications and would overload the picture. The other is to first clarify and reduce the context $\mathbb{C}_{\text{imp}}(\mathbb{K})$ and then write only one implication to each join-irreducible lattice element. This would simplify the labelling to only four implications. The disadvantage then however is that it is difficult to reconstruct the extents of the unreduced context. The bottom element, for example, would get no label at all. Its extent is the (nontrivial) set of **all-conditional** implications, i.e., those that hold under *all* conditions.

The problem comes from the fact that we have two different closure operators on implications simultaneously. Each set $\mathcal{L} \subseteq \text{Imp}(M)$ of implications generates both an *implicational theory* and an *extent of* $\mathbb{C}_{imp}(\mathbb{K})$. The latter always contains the first, since the attribute intents of $\mathbb{C}_{\text{imp}}(\mathbb{K})$ are implicational theories. Our labelling was chosen in such a way that each extent is generated as an implicational theory by the labels attached to that concept and to its subconcepts. This makes it sometimes necessary to label join-reducible elements, too. As an implicational theory, a join-irreducible extent is not necessarily generated by the union of the extents of its lower neighbours.

Nevertheless such a labelling is rather canonical. The bottom element is labelled with the stem base[3] of the all-conditional implications. We then can move up from the bottom element and label each node with the stem base relative to the union of the extents below it.

[3] We have simplified implications when possible.

We have claimed above – without giving a proof – that each extent of the lattice of conditional attribute implications is an implicational theory. This information was needed to ensure the existence of the (relative) stem bases used for the labelling. In fact, note that for each concept $(\mathcal{F}, D)$ of this lattice the extent $\mathcal{F}$ consists of precisely those implications that hold in the subposition of the contexts

$$(G, M, Y_{12b}), \quad b \in D,$$

where

$$(g, m) \in Y_{12b} :\Longleftrightarrow (g, m, b) \in Y.$$

The hierarchical structure of the implication families also indicates how to design a *conditional attribute exploration.* Such an exploration could start with an exploration of the all-conditional implications, i.e., with the bottom element of Figure 2, and then follow a linear extension to explore the remaining elements of the lattice of conditional attribute implications. The questions answered in the affirmative then correspond to the implications of the labelling. However, the data set we have chosen as our example is too trivial to justify such an exploration.

4 Attributional Condition Implications

If I do not get a meal with meat but no pork, I ask for a vegetarian dish.

The conditional implications treated above may be insufficient to express such statements (even if we include negated conditions). The sentence asks for a relation between *conditions* (and their negations, an aspect which we ignore here).

The notion of a triadic context has a sixfold symmetry: we may arbitrarily interchange the rôles of objects, attributes, and conditions. For example, if we exchange attributes and conditions in the previous section, we obtain what one might then call the **attributional condition implications.** Repeating the context construction above, we get the concept lattice shown in Figure 3.

Reading the diagram from the left, we see that on each day we can get a meal without special properties at Mommsenstraße and Bergstraße, and a vegetarian meal at Bergstraße, Reichenbachstraße, and Klinikum, et cetera. We also explain the meaning of the rightmost implication, 🐖, □→ 🍷. It says that whenever a meal with property 🐖 and a meal with property □ is served at Siedepunkt or at Klinikum, then the same cafeteria will also serve a meal with 🍷 on the same day. This is trivially true because the premise is never fulfilled. But note that the implication is false for each of the three other cafeterias: on Monday, both Bergstraße and Mommsenstraße serve meals with 🐖 and with □, but no meal with 🍷. For Reichenbachstraße this is so on Tuesday.

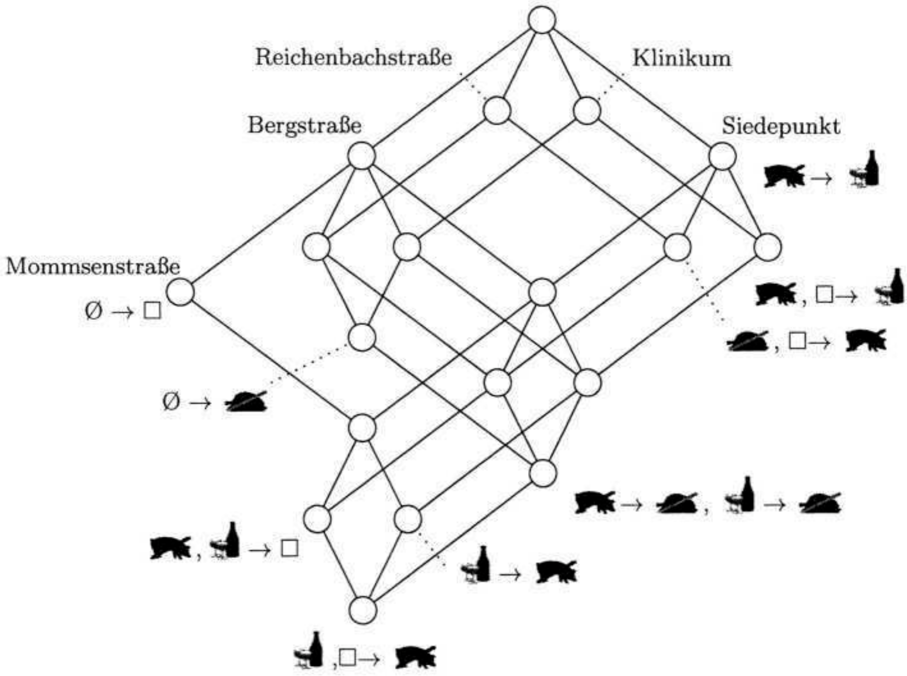


Fig. 3. The lattice of attributional condition implications.

As above, we can describe each concept extent of a concept with intent $N \subset M$ as the set of attribute implications of a subposition, namely that of the contexts

$$(G, B, Y_{1m3}), \quad m \in N,$$

where

$$(g, b) \in Y_{1m3} :\Longleftrightarrow (g, m, b) \in Y.$$

An interesting question which we cannot answer at present is how the conditional attribute implications and the attributional condition implications are connected. Clearly these sets are not completely independent. For example, let $m \in M$ and $b \in B$. The conditional implication $\varnothing \overset{b}{\to} m$ does *not* hold if there is some object $g \in G$ such that $(g, m, b) \notin Y$. But this is also necessary and sufficient for the attributional implication $\varnothing \overset{m}{\to} b$ not to hold. Thus these two implications are equivalent. A general characterization of the interplay is an open problem.

5 Attribute × Condition Implications

We get, as already explained, six lattices of implication families, two of which are shown in Figures 2 and 3.

Are there any open questions? More precisely, do these implications cover all interesting implications we may want to consider for the given data? This is not the case or, at least, not obviously so. The implications studied so far cannot express that

> *whenever a vegetarian meal is served at Siedepunkt, there is a meal without pork at Klinikum.*

We therefore investigate another generalization of Biedermann's implications. An **attribute×condition implication** is an expression of the form

$$R \to S,$$

where R and S are subsets of $M \times B$. The example above gives two singleton sets $R := \{(\text{Siedepunkt}, 🥦)\}, S := \{(\text{Klinikum}, 🐖)\}$ and reads (in simplified notation) as

$$(\text{Siedepunkt}, 🥦) \to (\text{Klinikum}, 🐖).$$

It *holds* in the triadic interpretation of Figure 1 since in fact every object (=day) having the attribute Siedepunkt under the condition 🥦 also has the attribute Klinikum under the condition 🐖. In this sense these are precisely the attribute implications of the formal context

$$(G, M \times B, Y_{1.23}) \qquad \text{with } (g,(m,b)) \in Y_{1.23} :\Longleftrightarrow (g,m,b) \in Y.$$

We can therefore describe this family of implications by means of the stem base of $(G, M \times B, Y_{1.23})$.

But this would be rather redundant. Even in our small example this stem base has 21 implications. Not all of them are actually needed, since we already have a considerable amount of information about the attribute×condition implications that hold in (G, M, B, Y), because we have computed the implication families in Figures 2 and 3. From each conditional attribute implication

$$R \xrightarrow{C} S$$

we can infer the attribute×condition implications

$$R \times \{c\} \to S \times \{c\} \quad \text{for all } c \in C,$$

and similarly each attributional condition implication $C \xrightarrow{N} D$ yields the attri”-bu”-te×condition implication $\{n\} \times C \to \{n\} \times D$ for all $n \in N$. Clearly, the backward inference (from corresponding sets of implications to attribute × conditional implications) is also possible.

Therefore, in the presence of the two lattices of implication families given in the preceding sections, what we would like to get is a basis of attribute×condition implications relative to conditional attribute and attributional condition implications. Although of different nature, implications of the latter two types can be easily transformed into sets of attribute×conditional implications as discussed above, which makes the defintion of the relative basis applicable. We computed three relative bases of attribute×condition implications: relative to conditional attribute implications (14 implications), relative to attributional condition implications (17 implications), and relative to the union of the two (11 implications). The smallest base is shown in Figure 4.

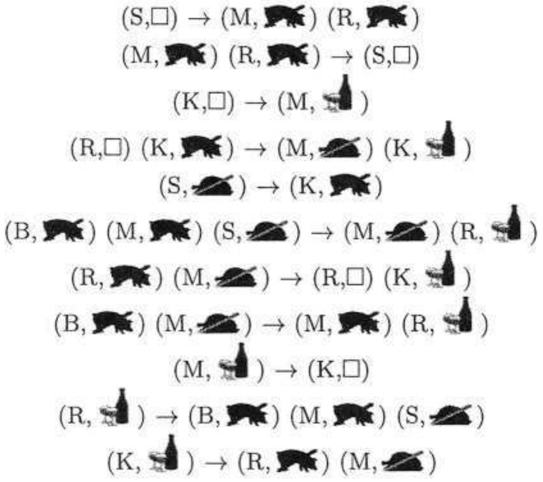

(S,□) → (M,) (R,)
(M,) (R,) → (S,□)
(K,□) → (M,)
(R,□) (K,) → (M,) (K,)
(S,) → (K,)
(B,) (M,) (S,) → (M,) (R,)
(R,) (M,) → (R,□) (K,)
(B,) (M,) → (M,) (R,)
(M,) → (K,□)
(R,) → (B,) (M,) (S,)
(K,) → (R,) (M,)

Fig. 4. A relative base for the attribute×condition implications.

If one is interested only in attribute×condition implications, it might be more appropriate to compute directly the stem base of $(G, M \times B, Y_{1.23})$, since this is the most compact way of representing all such implications and whatever techniques we use we cannot hope to achieve any reduction here. However, attribute×condition implications are still not the only kind of potentially interesting information. In our example, object×attribute implications would make some sense:

> *I can get in Reichenbachstraße on Tuesday whatever I can get in Bergstraßte on Monday.*

The information contained in object-attribute implications is partially contained in the all-conditional attribute implications. The results above may therefore help to find first examples of such implications.

Of course, object-attribute implications can be handled in exactly the same manner as we have demonstrated above. It remains open how to integrate all these implication notions in such a way that a reconstruction of the triconceptual structure from the implications becomes possible.

6 Conclusion

For triadic contexts there are several families of implications that are of interest. We have shown how these can be generated and displayed in a compact way. The families are interrelated, but their dependencies have not yet been fully clarified.

References

1. Klaus Biedermann. *A Foundation of the Theory of Trilattices.* Shaker Verlag, Aachen 1998, ISBN 3-8265-4028-X.
2. Frithjof Dau and Rudolf Wille. *On the modal unterstanding of triadic contexts.* In: R. Decker, W. Gaul (eds.): Classification and Information Processing at the Turn of the Millenium. Proceedings of the 23rd Annual Conference of the GfKl, University of Bielefeld, March 10-14, 1999. Series: Studies in Classification, Data Analysis, and Knowledge Organization. ISBN: 3-540-67589-2. Springer-Verlag, Heidelberg-Berlin, 2000
3. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis – Mathematical Foundations.* Springer Verlag Berlin-Heidelberg-New York, 1999.
4. J.-L. Guigues and Vincent Duquenne. *Families minimales d'implications informatives resultant d'un tableau de données binaires.* Math. Sci. Humaines 95 (1986).
5. Fritz Lehmann and Rudolf Wille.*A triadic approach to formal concept analysis.* In: G.Ellis, R.Levinson, W.Rich, J.F. Sowa (eds.): Conceptual structures: applications, implementation and theory. LNAI 954. Springer, Berlin-Heidelberg-New York, 32–43, 1995.
6. Gerd Stumme. *Attribute exploration with background implications and exceptions.* In: H.-H. Bock, W.Polasek (eds.): Data analysis and information systems. Springer, Berlin-Heidelberg-New York, 457–469, 1996.
7. Rudolf Wille. *The basic theorem of triadic concept analysis.* Order 12, 149–158, 1995.

Exploring Relational Structures Via $\mathcal{FLE}$

Sebastian Rudolph*

Institute of Algebra
Department of Mathematics and Natural Sciences
Dresden University of Technology
Germany
rudolph@math.tu-dresden.de

Abstract. Designing ontologies and specifying axioms of the described domains is an expensive and error-prone task. Thus, we propose a method originating from Formal Concept Analysis which uses empirical data to systematically generate hypothetical axioms about the domain, which are represented to an ontology engineer for decision.
In this paper, we focus on axioms that can be expressed as entailment statements in the description logic $\mathcal{FLE}$. The proposed technique is an incremental one, therefore, in every new step we have to reuse the axiomatic information acquired so far. We present a sound and complete deduction calculus for $\mathcal{FLE}$ entailment statements.
We give a detailed description of this multistep algorithm including a technique called empirical attribute reduction and demonstrate the proposed technique using an example from mathematics.
We give a completeness result on the explored information and address the question of algorithm termination. Finally, we discuss possible applications of our method.

1 Introduction

When designing systems for knowledge representation and exchange (such as expert systems, semantic web applications, ontologies in general, etc.) one central task is to specify not only the basic terms used to characterize the entities of the described domain but also the logical interrelationships between them. This information (called domain axioms or rules) encodes the background or world knowledge and enables automatic reasoning about the domain.

Since the system's knowledge has to match reality, this specification task can not be carried out fully automatically, unless one has already a complete representation of the part of the world to be described. Otherwise human assistance is necessary. Nevertheless, also incomplete data about reality maybe extremely helpful in order to reduce the set of possible axioms in advance (by assuming their consistency with the data).

In this paper (extending our former publication [13]) we present an algorithm that helps to determine all domain axioms of a certain logical shape by successively presenting questions to an expert. This is done in a way, such that no

* Supported by DFG/Graduiertenkolleg.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 196–212, 2004.

redundant questions will be asked. Additional information (concerning entities or a priori known rules of the domain) are taken into account.

Section 2 will shortly recall some notions of formal concept analysis (formal context, implication, stem base) as far as they are needed for an understanding of the attribute exploration algorithm.

In Section 3, based on Description Logic (DL) a class of concept descriptions ($\mathcal{FLE}$) is defined together with an extensional semantic using binary power context families. Definitions of entailment and equivalence of that formulae with respect to a fixed semantics are discussed.

In Section 4, we define a special kind of formal contexts that can be constructed on the basis of a binary power context family from a set of DL-formulae. We observe that implications within such a formal context correspond to valid DL entailment statements.

The algorithm, that consists of a sequence of exploration steps is described in Section 5: initialization, the actual exploration step yielding a stem base $\mathfrak{B}_i$, and how the stem base can be used to determine the attribute set and background knowledge for the next exploration step.

Section 6 discusses how the validity of an arbitrary entailment statement between concept descriptions from $\mathcal{FLE}_i$ can be decided using just the stem bases $\mathfrak{B}_0, \dots, \mathfrak{B}_i$ obtained by the exploration process.

Section 7 addresses the question under which conditions the proposed algorithm terminates, i.e., in which cases a complete information acquisition is achieved. We will demonstrate how any $\mathcal{FLE}$ formula can then be decided based on the explored knowledge.

In Section 8, we apply the presented algorithm to an example from basic mathematics.

Concluding, in Section 9 we discuss how this algorithm can be applied e.g. for generating and refining ontologies.

2 Attribute Exploration

Here, we will introduce notions from Formal Concept Analysis relevant for this theory. For a comprehensive introduction into FCA cf. [6].

Definition 1. *A* FORMAL CONTEXT $\mathbb{K} := (G, M, I)$ *consists of two arbitrary sets G (the elements of which are called objects) and M (the elements of which are called attributes) and a relation $I \subseteq G \times M$. The incidence gIm for $g \in G$ and $m \in M$ is read as "object g has attribute m".*

Definition 2. *LetM be an arbitrary set. IfA andB are two sets with $A, B \subseteq M$ we will call the pair (A, B) an* IMPLICATION *on M. To support intuition we will write it as $A \rightarrow B$ in the sequel. We say an implication* HOLDS *for an attribute set C, iff from $A \subseteq C$ follows $B \subseteq C$. Moreover, an implication* HOLDS *in a formal context $\mathbb{K} = (G, M, I)$ iff it holds for all its object intents $g^I := \{m \in M \mid gIm\}$.*

Given a set $A \subseteq M$ and a set $\mathfrak{J}$ of implications on M, we write $A^{\mathfrak{J}}$ for the smallest subset ofM which

- *contains A and*
- *fulfills all implications from* $\mathfrak{J}$.[1]

Let $imp(\mathbb{K})$ *denote the set of all implications holding in* $\mathbb{K}$. *A set of implications* $\mathfrak{B}$ *is called* IMPLICATION BASE *of* $\mathbb{K}$ *iff it is*

- *complete, i.e.,* $A^{\mathfrak{B}} = A^{imp(\mathbb{K})}$ *for all* $A \subseteq M$ *and*
- *irredundant, i.e., for every implication* $i \in \mathfrak{B}$ *there is an* $A \subseteq M$ *with* $A^{\mathfrak{B}\setminus\{i\}} \neq A^{imp(\mathbb{K})}$.

Guigues and Duquenne [8] found a characterization of a canonical minimal implication base - the so called *stem base.* Ganter's attribute exploration algorithm [5] is an interactive method to determine the implication base of a formal context not entirely known in the beginning. The algorithm systematically presents potential implications (i.e., such ones that do not contradict the known part of the context) asking for their overall validity. A domain expert then has to decide: either (s)he confirms the implication - in that case it will be incorporated into the implication base - or he denies it - then he has to state a counterexample object which will be added to the considered context. This process continues, until the implications of the (still partial) context are just those mediated by the generated implication base. Figure 1 shows a scheme of the algorithm.

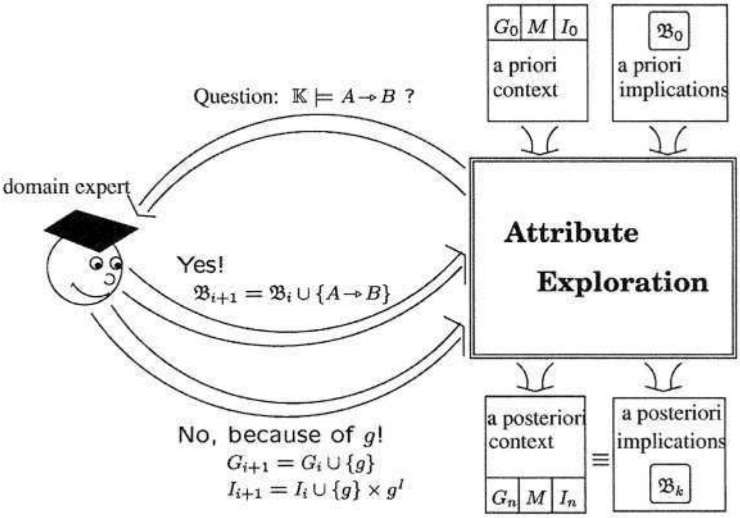

G_0 M I_0
a priori context
$\mathfrak{B}_0$
a priori implications
Question: $\mathbb{K} \models A \rightarrow B$?
domain expert
Attribute Exploration
Yes!
$\mathfrak{B}_{i+1} = \mathfrak{B}_i \cup \{A \rightarrow B\}$
No, because of g!
$G_{i+1} = G_i \cup \{g\}$
$I_{i+1} = I_i \cup \{g\} \times g'$
a posteriori context
G_n M I_n
$\equiv$
a posteriori implications
$\mathfrak{B}_k$

Fig. 1. Scheme of Ganter's attribute exploration algorithm.

[1] Since those two requirements are preserved under intersection, the existence of a smallest such set is assured. Moreover, note that the operation $(.)^{\mathfrak{J}}$ is a closure operator on *M.* Note also that, given *A* and $\mathfrak{J}$, the closure can be calculated in linear time (cf. [4]).

3 The Language $\mathcal{FLE}$: Syntax and Semantics

In this section we will introduce the Description Logic $\mathcal{FLE}$. We will just introduce the notions needed for our purposes, for a comprehensive overview see [2]. At first we will define the set $\mathcal{FLE}$ of all concept descriptions:

Definition 3. *Let $M_\mathcal{C}$, $M_\mathcal{R}$ be arbitrary finite sets, the elements of which we will call concept*[2] *names and role names, respectively. By $\mathcal{FLE}(M_\mathcal{C}, M_\mathcal{R})$ (or shortly: $\mathcal{FLE}$ if there is no danger of confusion) we denote the set of formulae (also called* CONCEPT DESCRIPTIONS) *inductively defined as follows:*

$$\begin{aligned} M_\mathcal{C} \cup \{\top, \bot\} &\subseteq \mathcal{FLE}, \\ \varphi, \psi \in \mathcal{FLE} &\Rightarrow \varphi \sqcap \psi \in \mathcal{FLE}, \\ \varphi \in \mathcal{FLE}, r \in M_\mathcal{R} &\Rightarrow \exists r.\varphi \in \mathcal{FLE}, \\ \varphi \in \mathcal{FLE}, r \in M_\mathcal{R} &\Rightarrow \forall r.\varphi \in \mathcal{FLE}. \end{aligned}$$

By $\mathcal{FLE}_n$ we denote the set of all concept descriptions from $\mathcal{FLE}$ with role depth of at most n.

Next, we will define what is an interpretation of $\mathcal{FLE}$. Sticking to the way of defining relational structures usual in FCA (see also [16]) we call it binary power context family. The common definitions in DL and modal logics (see e.g. [3]) are just syntactic variants thereof.

Definition 4. *A* BINARY POWER CONTEXT FAMILY *on a set Δ, called the* UNIVERSE, *with $\Delta \neq \emptyset$ is a pair $(\mathbb{K}_\mathcal{C}, \mathbb{K}_\mathcal{R})$ consisting of the formal contexts $\mathbb{K}_\mathcal{C} := (G_\mathcal{C}, M_\mathcal{C}, I_\mathcal{C})$ and $\mathbb{K}_\mathcal{R} := (G_\mathcal{R}, M_\mathcal{R}, I_\mathcal{R})$ with $G_\mathcal{C} = \Delta$ and $G_\mathcal{R} = \Delta \times \Delta$.*

As we know from the definition of formal context, $M_\mathcal{C}$ and $M_\mathcal{R}$ are arbitrary sets and $I_\mathcal{C} \subseteq G_\mathcal{C} \times M_\mathcal{C}$ as well as $I_\mathcal{R} \subseteq G_\mathcal{R} \times M_\mathcal{R}$.

Definition 5. *The semantical mapping $[\![.]\!]_{\vec{\mathbb{K}}} : \mathcal{FLE}(M_\mathcal{C}, M_\mathcal{R}) \to \mathcal{P}(\Delta)$ for a binary power context family $\vec{\mathbb{K}}$ on a universe Δ with attribute sets $M_\mathcal{C}, M_\mathcal{R}$ is defined recursively:*

$$\begin{aligned} [\![\top]\!]_{\vec{\mathbb{K}}} &:= \Delta, \\ [\![\bot]\!]_{\vec{\mathbb{K}}} &:= \emptyset, \\ [\![m]\!]_{\vec{\mathbb{K}}} &:= m^{I_\mathcal{C}} \text{ for } m \in M_\mathcal{C}, \\ [\![\varphi \sqcap \psi]\!]_{\vec{\mathbb{K}}} &:= [\![\varphi]\!]_{\vec{\mathbb{K}}} \cap [\![\psi]\!]_{\vec{\mathbb{K}}}, \\ [\![\exists r.\varphi]\!]_{\vec{\mathbb{K}}} &:= \{x \in \Delta \mid \exists y : (x,y) \in r^{I_\mathcal{R}} \wedge y \in [\![\varphi]\!]_{\vec{\mathbb{K}}}\} \quad \text{for } r \in M_\mathcal{R}, \\ [\![\forall r.\varphi]\!]_{\vec{\mathbb{K}}} &:= \{x \in \Delta \mid \forall y : (x,y) \in r^{I_\mathcal{R}} \to y \in [\![\varphi]\!]_{\vec{\mathbb{K}}}\} \quad \text{for } r \in M_\mathcal{R}. \end{aligned}$$

[2] Whenever in this article we use the term *concept* we refer to the notion used in Description Logic. If we want to refer to the meaning used in Formal Concept Analysis (FCA) we use *formal concept.*

Verbally, we have defined an extensionsal semantics, assigning to every concept description all entities of the universe fulfilling that description.

Furthermore, we say a formula φ is VALID IN $\vec{\mathbb{K}}$ (which we denote by $\vec{\mathbb{K}} \models \varphi$), iff $[\![\varphi]\!]_{\vec{\mathbb{K}}} = \Delta$. A formula φ ENTAILS a formula ψ IN $\vec{\mathbb{K}}$ (write: $\varphi \models_{\vec{\mathbb{K}}} \psi$), iff $[\![\varphi]\!]_{\vec{\mathbb{K}}} \subseteq [\![\psi]\!]_{\vec{\mathbb{K}}}$. A formula φ ENTAILS a formula ψ in general (write: $\varphi \models \psi$), iff $[\![\varphi]\!]_{\vec{\mathbb{K}}} \subseteq [\![\psi]\!]_{\vec{\mathbb{K}}}$ for all binary power context families $\vec{\mathbb{K}}$ with appropriate signature. Two formulae φ and ψ are called $\vec{\mathbb{K}}$–EQUIVALENT, iff $\varphi \models_{\vec{\mathbb{K}}} \psi$ and $\psi \models_{\vec{\mathbb{K}}} \varphi$ (write: $\varphi \equiv_{\vec{\mathbb{K}}} \psi$). They are EQUIVALENT, iff $\varphi \models \psi$ and $\psi \models \varphi$ (write: $\varphi \equiv \psi$).

Let $C = \{c_1, \ldots, c_n\}$ be a finite set of $\mathcal{FLE}$ concept descriptions. Then the new concept description $c_1 \sqcap \ldots \sqcap c_n$ will be abbreviated by $\bigsqcap C$. Moreover, let $\bigsqcap\{c\} = c$ and $\bigsqcap\emptyset = \top$.

Finally, note that $\forall r.(c_1 \sqcap c_2)$ and $\forall r.c_1 \sqcap \forall r.c_2$ are equivalent for any concept descriptions c_1, c_2 and any role r. So, for every concept description $c \in \mathcal{FLE}$ there is an equivalent "sibling" $\tilde{c} \in \mathcal{FLE}$ where in no subformula $\forall r.\varphi$ the φ is a conjunction. In the sequel, we assume any formula we deal with to be normalized in this way.

The following abbreviations for sets of $\mathcal{FLE}$ concept descriptions have been found both intuitive and useful:

$$[\exists r]A := \begin{cases} \{\bot\} \text{ if } \bot \in A, \\ \{\exists r.\bigsqcap A\} \text{ otherwise} \end{cases}$$

and

$$[\forall r]A := \{\forall r.a \mid a \in A\}.$$

By restricting to a reduced amount of logic features (omitting disjunction and negation) we obtain a class of propositions that can be managed algorithmically in practice and - as we suppose (see also [15]) - still comprises the majority of human conceptual thinking.

4 $\mathcal{FLE}$-Contexts

As our aim is to use the exploration algorithm in order to collect information expressable by $\mathcal{FLE}$ statements it is just natural to define a kind of formal context, where the attributes are arbitrary $\mathcal{FLE}$ concept descriptions:

Definition 6. *Given a binary power context family $\vec{\mathbb{K}} = (\mathbb{K}_{\mathcal{C}}, \mathbb{K}_{\mathcal{R}})$ on a universe Δ and a set $M \subseteq \mathcal{FLE}(M_{\mathcal{C}}, M_{\mathcal{R}})$, the corresponding $\mathcal{FLE}$-CONTEXT is defined in the following way:*

$$\mathbb{K}_{\mathcal{FLE}}(M) := (\Delta, M, I) \text{ with } \delta I m :\Leftrightarrow \delta \in [\![m]\!]_{\vec{\mathbb{K}}}.$$

Formal contexts where attributes are DL-formulae and the incidence relation is defined via validity have been described by Prediger in [12]. While she was aiming at extending a context by “interesting” new attributes, we want to explore $\mathcal{FLE}$-contexts and therefore more attention at the choice of attributes is required as we will see in the sequel.

Now, what does validity of an implication in such an $\mathcal{FLE}$-context mean from the point of view of DL? Suppose

$$\{m_1, \dots, m_k\} \rightarrow \{m_{k+1}, \dots, m_l\}$$

is an implication valid in $\mathbb{K}_{\mathcal{FLE}}$. The following theorem shows that this is equivalent to the validity of the entailment statement

$$\sqcap\{m_1, \dots, m_k\} \models_{\vec{\mathbb{K}}} \sqcap\{m_{k+1}, \dots, m_l\}.$$

Theorem 1. *Let $\vec{\mathbb{K}}$ be a binarypower contextfamily, $M \subseteq \mathcal{FLE}$ and $A, B \subseteq M$. Then the implication $A \rightarrow B$ is valid in $\mathbb{K}_{\mathcal{FLE}}(M)$ iff $\sqcap A \models_{\vec{\mathbb{K}}} \sqcap B$.*

Proof. $\mathbb{K}_{\mathcal{FLE}}(M) \models A \rightarrow B$ iff for all $\delta \in \Delta$ from $A \subseteq \delta^I$ follows $B \subseteq \delta^I$. This is the case iff $\bigcap\{a^I \mid a \in A\} \subseteq \bigcap\{b^I \mid b \in B\}$, which due to the definition of I is equivalent to $\bigcap\{[\![a]\!]_{\vec{\mathbb{K}}} \mid a \in A\} \subseteq \bigcap\{[\![b]\!]_{\vec{\mathbb{K}}} \mid b \in B\}$ and thus also to $[\![\sqcap A]\!]_{\vec{\mathbb{K}}} \subseteq [\![\sqcap B]\!]_{\vec{\mathbb{K}}}$. □

So this is the way how implications gained by an exploration process of an $\mathcal{FLE}$-context can be reinterpreted as entailment rules.

5 Successive Exploration

In the past there have been some approaches to apply the FCA exploration technique to a logic more expressive than propositional logic. Zickwolff used Ganter’s algorithm to determine the first order Horn theory of a certain domain in [17].

In this section we describe the multistep exploration algorithm in detail.

At first, we have to stipulate $M_{\mathcal{C}}$ and $M_{\mathcal{R}}$ - the concepts and roles which’s domain specific interrelationships we are interested in.

Next we may provide some empirical data by naming known entities $G \subseteq \Delta$ of the considered universe and stating their attributes.

Moreover, we can input axiomatic information about the domain by stating $\mathcal{FLE}$ entailment statements already known to hold.

So, we start the exploration sequence with the context $\mathbb{K}_0 = (G_0, M_0, I_0)$ where
$G_0 := G$,
$M_0 := M_{\mathcal{C}} \cup \{\bot\}$, and
$I_0 := I_{\mathcal{C}} \cap G \times M_{\mathcal{C}}$.

The exploration is carried out as described in Section 2. Every implication $\{m_1, \dots, m_k\} \rightarrow \{m_{k+1}, \dots, m_l\}$ being presented to the expert has to be interpreted in the following way: "Do all entities from the universe that fulfill the concept description $m_1 \sqcap \dots \sqcap m_k$ also fulfill the concept description $m_{k+1} \sqcap \dots \sqcap m_l$?" The expert either confirms this, or provides an entity that violates this condition. The result of this first exploration step is the stem base $\mathfrak{B}_0$.

When one such step (say: the one generating $\mathfrak{B}_{i-1}$) has finished, the next one has to be prepared:

First, the attribute set M_i is generated as follows:

$$\begin{aligned} M_i := {} & M_C \cup \{\bot\} \\ & \cup \{\exists r. \textstyle\bigsqcap A \mid r \in M_\mathcal{R}, A = A^{\mathfrak{B}_{i-1}} \subseteq M_{i-1} \setminus \{\bot\}\} \\ & \cup \{\forall r.m \mid r \in M_\mathcal{R}, m \in M_{i-1}\}. \end{aligned}$$

This choice of the attributes is motivated by the purpose to keep the set of attributes small (which is essential for the exploration algorithm since its worst case complexity increases exponentially with the number of attributes involved) while preserving the completeness we deal with in Section 6.

Note that for every attribute $m \in M_{i-1}$ we find an attribute $\widetilde{m} \in M_i$ with $m \equiv_{\mathfrak{K}} \widetilde{m}$ by using the function $\varphi_i : \mathcal{FLE} \to \mathcal{P}(\mathcal{FLE})$ defined as follows:

$$\begin{aligned} \varphi_i(c) &:= \{c\} \text{ for } c \in M_C \cup \{\bot\} \\ \varphi_i(\forall r.c) &:= [\forall r]\varphi_{i-1}(c) \\ \varphi_i(\exists r.c) &:= [\exists r](\varphi_{i-1}(c))^{\mathfrak{B}_{i-1}} \\ \varphi_i(\textstyle\bigsqcap C) &:= \bigcup\{\varphi_i(c) \mid c \in C\} \end{aligned}$$

It is easy to see that for $m \in M_{i-1}$ the term $\varphi_i(m)$ yields a singleton set. Now we take the only element of this set as our representative $\widetilde{m}$. The facts $\widetilde{m} \in M_i$ and $m \equiv_{\mathfrak{K}} \widetilde{m}$ are immediate consequences of the following proofs.

Theorem 2. *Let $c \in \mathcal{FLE}_i$. Then $\varphi_i(c) \subseteq M_i$.*

Proof. We show this using induction on the role depth of c considering four cases:

- $c \in M_C \cup \{\bot\}$. Then, by definition, $\{c\} \subseteq M_i$.
- $c = \forall r.\widetilde{c}$. As our induction hypothesis assures, we have $\varphi_{i-1}(\widetilde{c}) \subseteq M_{i-1}$ and due to the definition of M_i this directly implies $[\forall r]\varphi_{i-1}(\widetilde{c}) \subseteq M_i$.
- $c = \exists r.\widetilde{c}$. As induction hypothesis we have $\varphi_{i-1}(\widetilde{c}) \subseteq M_{i-1}$. But then we have also $\exists r.\bigsqcap \varphi(\widetilde{c}, i-1)^{\mathfrak{B}_{i-1}} \in M_i$, as a look at the constructive definition of M_i immediately shows.
- $c = \bigsqcap \widetilde{C}$. W.l.o.g. we presuppose there is no conjunction outside the quantifier range in any $\widetilde{c} \in \widetilde{C}$. So we have $\varphi_i(\widetilde{c}) \subseteq M_i$, due to the three cases above. □

Lemma 1. *For any $A \subseteq M_i$ we have $\sqcap A \equiv_{\mathbb{K}} \sqcap A^{\mathfrak{B}_i}$.*

Proof. From Theorem 1 we know that for every entity $\delta \in \Delta$ the set of its attributes $M \in \mathcal{FLE}_i$ fulfills all implications from $\mathfrak{B}_i$. Hence, when considering only those δ having all attributes from A, every one of them must even have every attribute from $A^{\mathfrak{B}_i}$, since this is the smallest attribute set containing A and satisfying $\mathfrak{B}_i$. Therefore we have $\bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A\} \subseteq \bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A^{\mathfrak{B}_i}\}$.

On the other hand we have trivially $\bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A^{\mathfrak{B}_i}\} \subseteq \bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A\}$, for the left hand side intersection contains at least all sets from the right hand side intersection. So, finally we get

$$\llbracket\sqcap A\rrbracket_{\mathbb{K}} = \bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A\} = \bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in A^{\mathfrak{B}_i}\} = \llbracket\sqcap A^{\mathfrak{B}_i}\rrbracket_{\mathbb{K}}.$$

□

Theorem 3. *Let $c \in \mathcal{FLE}_i$. Then $c \equiv_{\mathbb{K}} \sqcap\varphi_i(c)$.*

Proof. We show this again via induction on the role depth:

- $c \in M_C \cup \{\bot\}$. Then we have $\llbracket c\rrbracket_{\mathbb{K}} = \llbracket\sqcap\{c\}\rrbracket_{\mathbb{K}}$.
- $c = \forall r.\tilde{c}$. By induction hypothesis we have $\llbracket\tilde{c}\rrbracket_{\mathbb{K}} = \llbracket\sqcap\varphi_{i-1}(\tilde{c})\rrbracket_{\mathbb{K}}$ implying $\llbracket\forall r.\tilde{c}\rrbracket_{\mathbb{K}} = \llbracket\sqcap[\forall r]\varphi_{i-1}(\tilde{c})\rrbracket_{\mathbb{K}}$ which by definition equals $\llbracket\sqcap\varphi_i(\forall r.\tilde{c})\rrbracket_{\mathbb{K}}$.
- $c = \exists r.\tilde{c}$. By induction hypothesis we have $\llbracket\tilde{c}\rrbracket_{\mathbb{K}} = \llbracket\sqcap\varphi_{i-1}(\tilde{c})\rrbracket_{\mathbb{K}}$, and since $\llbracket\sqcap\varphi_{i-1}(\tilde{c})\rrbracket_{\mathbb{K}} = \llbracket\sqcap\varphi_{i-1}(\tilde{c})^{\mathfrak{B}_{i-1}}\rrbracket_{\mathbb{K}}$ due to Lemma 1 we have $\llbracket\exists r.\tilde{c}\rrbracket_{\mathbb{K}} = \llbracket\exists r.\sqcap(\varphi_{i-1}(\tilde{c}))^{\mathfrak{B}_{i-1}}\rrbracket_{\mathbb{K}}$ which by definition equals $\llbracket\sqcap\varphi_i(\exists r.\tilde{c})\rrbracket_{\mathbb{K}}$.
- $c = \sqcap\tilde{C}$. Again we can preassume no conjunction outside the quantifier range in any $\tilde{c} \in \tilde{C}$. Then $\llbracket\sqcap\tilde{C}\rrbracket_{\mathbb{K}} = \bigcap\{\llbracket\tilde{c}\rrbracket_{\mathbb{K}} \mid \tilde{c} \in \tilde{C}\} = \bigcap\{\llbracket\sqcap\varphi_i(\tilde{c})\rrbracket_{\mathbb{K}} \mid \tilde{c} \in \tilde{C}\}$ because of the cases shown before. Now, this is obviously the same as $\bigcap\{\llbracket m\rrbracket_{\mathbb{K}} \mid m \in \varphi_i(\tilde{c}), \tilde{c} \in \tilde{C}\} = \llbracket\sqcap(\bigcup\{\varphi_i(\tilde{c}) \mid \tilde{c} \in \tilde{C}\})\rrbracket_{\mathbb{K}}$. □

This allows us to reuse all implications from the former exploration step as input for the next one: We simply add $\varphi_i(\sqcap A) \rightarrow \varphi_i(\sqcap B)$ for all $A \rightarrow B \in \mathfrak{B}_{i-1}$ to the background knowledge.

But there is more a priori knowledge that can be extracted from $\mathfrak{B}_{i-1}$. Exploiting the deduction calculus presented in the appendix, we can augment our a priori information even further. So we add:

- $\{\bot\} \rightarrow M_i$ (due to the $\mathcal{C}$ rule),

- $\{\exists r.\sqcap A\} \to \{\exists r.\sqcap B\}$ for all $B \subseteq A$ (as a consequence of the $\mathcal{ID}$, $\mathcal{PE}$, and $\exists\mathcal{L}$ rules),
- $\{\exists r.\sqcap A, \forall r.b\} \to \{\exists r.\sqcap (A \cup \{b\})^{\mathfrak{B}_i}\}$ (because of the rules $\forall\mathcal{P}$ and $\exists\mathcal{L}$),
- $\{\forall r.a \mid a \in A\} \to \{\forall r.b \mid b \in B\}$ for all $A \to B \in \mathfrak{B}_i$ (justified by $\forall\mathcal{L}$).[3]

This algorithm can be carried out iteratively, thereby producing a sequence of implication bases $\mathfrak{B}_0, \mathfrak{B}_1 \ldots$. How these can be used for deciding "entailment queries" will be dealt with in the next chapter.

Baader presented a method for computing the subsumption hierarchy of all concept descriptions, that can be obtained by applying conjunction to concept names in [1]. His algorithm technically corresponds to our first exploration step (on the attribute set M_0) - but for the intended purpose: Baader suggests to let a DL subsumption algorithm take the role of the expert thus exploring the subsumptions valid for a given DL system, while we are aiming at finding information not yet being inherently present in the system.

6 Checking the Validity of an Entailment Statement

This section is dedicated to the question, which kind of information will be acquired after a certain step of the exploration algorithm. The answer is the following. Having explored a binary power context family until step i, we can decide for *any* entailment statement $c_1 \models_{\overrightarrow{\mathbb{K}}} c_2$ (with c_1, c_2 being arbitrary $\mathcal{FLE}$ concept descriptions with maximal role depth of at most i) whether it is valid in $\overrightarrow{\mathbb{K}}$ or not, using just the bases $\mathfrak{B}_0, \ldots, \mathfrak{B}_i$. In this sense the exploration algorithm is complete. The decision procedure works as follows: The function φ_i, defined in the preceding section, gives for any $c \in \mathcal{FLE}_i$ a set of attributes from M_i the conjunction of which yields a concept description that is $\overrightarrow{\mathbb{K}}$-equivalent to c. Therefore, as the following corollary shows the entailment of two concept descriptions c_1 and c_2 can simply be checked by exploiting the proposition

$$c_1 \models_{\overrightarrow{\mathbb{K}}} c_2 \iff \varphi_i(c_2)^{\mathfrak{B}_i} \subseteq \varphi_i(c_1)^{\mathfrak{B}_i}.$$

Corollary 1. *Let* $c_1, c_2 \in \mathcal{FLE}_i$. *Then* $c_1 \models_{\overrightarrow{\mathbb{K}}} c_2$ *iff* $\varphi_i(c_2)^{\mathfrak{B}_i} \subseteq \varphi_i(c_1)^{\mathfrak{B}_i}$.

Proof. Due to Theorem 3, $c_1 \models_{\overrightarrow{\mathbb{K}}} c_2$ is equivalent to $\sqcap\varphi_i(c_1) \models_{\overrightarrow{\mathbb{K}}} \sqcap\varphi_i(c_2)$. According to Theorem 2, we have $\varphi_i(c_1) \subseteq M_i$ and $\varphi_i(c_2) \subseteq M_i$. So via Theorem 1, this means the same as the validity of the implication $\varphi_i(c_1) \to \varphi_i(c_2)$ in $\mathbb{K}_i$. Obviously, this is equivalent to $\varphi_i(c_2)^{\mathfrak{B}_i} \subseteq \varphi_i(c_1)^{\mathfrak{B}_i}$. □

By checking entailment in both directions we also may find equivalences in $\overrightarrow{\mathbb{K}}$.

[3] Mark that the $\exists\mathcal{F}$-rule is already incorporated in the algorithm via the definition of $[\exists r]$. The $\mathcal{ID}$, $\mathcal{PE}$ and $\mathcal{MP}$ rules (being the usual implication deduction or Armstrong rules) do not need to be cared about since we are looking for implication bases, being just sets reduced with respect to the Armstrong rules.

7 Termination

At least from the theoretical point of view the question emerges, whether and under which circumstances the proposed algorithm terminates, i.e., all information expressable by $\mathcal{FLE}$ entailment statements has been acquired. We found that this is the case iff there is an $n \in \mathbb{N}$ such that the mapping $F_n : \{A^{\mathfrak{B}_n} \mid A \subseteq M_n\} \to \{B^{\mathfrak{B}_{n+1}} \mid B \subseteq M_{n+1}\}$ with $F_n(A) := (\varphi_{n+1}(\bigsqcap A))^{\mathfrak{B}_{n+1}}$ is a bijection between the $\mathfrak{B}_n$-closed subsets from M_n and the $\mathfrak{B}_{n+1}$-closed subsetsfrom M_{n+1}.

In the the following theorems we show some consequences of this property:

Theorem 4. *Let $\overrightarrow{\mathbb{K}}$ be a binarypower contextfamily with theproperty described above. Then*

1. *For any $B = B^{\mathfrak{B}_{n+1}} \subseteq M_{n+1}$ and $A = F_n^{-1}(B)$ we have $\bigsqcap A \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap B$.*
2. *For any $c \in \mathcal{FLE}_{n+1}$ we have $c \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap F_n^{-1}(\varphi_{n+1}(c))^{\mathfrak{B}_{n+1}}$.*

Proof.

Because $A \subseteq M_n$ we know $\bigsqcap A \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap \varphi_{n+1}(\bigsqcap A)$ due to Theorem 3. Applying Lemma 1 gives us $\bigsqcap \varphi_{n+1}(\bigsqcap A) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap(\varphi_{n+1}(\bigsqcap A))^{\mathfrak{B}_{n+1}}$. By definition of F_n we see that the right hand side of the equivalence is just $\bigsqcap F_n(A)$. Since $F_n(A) = B$, we are done.

The second proposition can then be proved as follows. We know $c \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap \varphi_{n+1}(c)$ by Theorem 3 and $\bigsqcap \varphi_{n+1}(c) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap(\varphi_{n+1}(c))^{\mathfrak{B}_{n+1}}$ by Lemma 1. From the first part of this theorem follows $\bigsqcap(\varphi_{n+1}(c))^{\mathfrak{B}_{n+1}} \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap F_n^{-1}(\varphi_{n+1}(c))^{\mathfrak{B}_{n+1}}$. □

This theorem provides a way to "shrink" an $\mathcal{FLE}_{n+1}$ formula to maximal quantifier depth n preserving its semantics. But - exploiting this fact - we can do even more: for any concept description $c \in \mathcal{FLE}$ we find an empirically equivalent concept description $\tilde{c} \in \mathcal{FLE}_n$ by applying the function $\pi : \mathcal{FLE} \to \mathcal{FLE}_n$ with[4]

$$d \mapsto d \text{ for all } d \in M_C \cup \{\bot\}$$

$$\exists r.d \mapsto \begin{cases} \bigsqcap[\exists r](\varphi_{n-1}(d))^{\mathfrak{B}_{n-1}} \text{ if } \exists r.d \in \mathcal{FLE}_n, \\ \bigsqcap F_n^{-1}([\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n})^{\mathfrak{B}_{n+1}} \text{ otherwise.} \end{cases}$$

$$\forall r.d \mapsto \begin{cases} \bigsqcap[\forall r]\varphi_{n-1}(d) \text{ if } \forall r.d \in \mathcal{FLE}_n, \\ \bigsqcap F_n^{-1}([\forall r]\varphi_n(\pi(d)))^{\mathfrak{B}_{n+1}} \text{ otherwise.} \end{cases}$$

$$\bigsqcap D \mapsto \bigsqcap\{\pi(d) \mid d \in D\}.$$

[4] In this notation, $(.)^{\mathfrak{B}}$ binds stronger than F_n^{-1}, φ_n, φ_{n-1}, $[\forall r]$, and $[\exists r]$.

Theorem 5. *Let $\overrightarrow{\mathbb{K}}$ be a binary power context family, $n \in \mathbb{N}$, and the corresponding F_n be a bijection. Then for any $c \in \mathcal{FLE}$ we have $\pi(c) \in \mathcal{FLE}_n$ and $\pi(c) \equiv_{\overrightarrow{\mathbb{K}}} c$.*

Proof. This proof will be done by induction on the quantifier depth. We have to consider the following cases:

- $c \in M_{\mathcal{C}} \cup \{\bot\}$.
  This is trivial: $c \equiv_{\overrightarrow{\mathbb{K}}} c = \pi(c)$.
- $c = \exists r.d \in \mathcal{FLE}_n$.

  Applying Theorem 3 and Lemma 1 yields $d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap \varphi_{n-1}(d) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap (\varphi_{n-1}(d))^{\mathfrak{B}_{n-1}}$, directly implying $\exists r.d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap [\exists r](\varphi_{n-1}(d))^{\mathfrak{B}_{n-1}} = \pi(c)$. Since $\varphi_{n-1}(d) \subseteq \mathcal{FLE}_{n-1}$, we also have $\pi(c) \subseteq \mathcal{FLE}_n$.
- $c = \exists r.d \notin \mathcal{FLE}_n$.
  By induction hypothesis, $d \equiv_{\overrightarrow{\mathbb{K}}} \pi(d)$. Theorem 3 and Lemma 1 give us $\pi(d) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap (\varphi_n(\pi(d)))^{\mathfrak{B}_n}$. From this we conclude $\exists r.d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap [\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n}$. Notice that the equivalence's right hand side is in M_{n+1} assured by Theorem 2 and the Definition of M_{n+1}. Applying Lemma 1 once more we get $\bigsqcap [\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n} \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap ([\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n})^{\mathfrak{B}_{n+1}}$ and by Theorem 4.1 we have $\bigsqcap ([\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n})^{\mathfrak{B}_{n+1}} \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap F_n^{-1}([\exists r](\varphi_n(\pi(d)))^{\mathfrak{B}_n})^{\mathfrak{B}_{n+1}} = \pi(c)$. So we have showed $c \equiv_{\overrightarrow{\mathbb{K}}} \pi(c)$. The application of F_n^{-1} assures $\pi(c) \in \mathcal{FLE}_n$.
- $c = \forall r.d \in \mathcal{FLE}_n$.

  Applying Theorem 3 yields $d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap \varphi_{n-1}(d)$, directly implying $\forall r.d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap [\forall r]\varphi_{n-1}(d) = \pi(c)$. Since $\varphi_{n-1}(d) \subseteq \mathcal{FLE}_{n-1}$, we also have $\pi(c) \subseteq \mathcal{FLE}_n$.
- $c = \forall r.d \notin \mathcal{FLE}_n$.

  By induction hypothesis, $d \equiv_{\overrightarrow{\mathbb{K}}} \pi(d)$. Theorem 3 gives us $\pi(d) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap \varphi_n(\pi(d))$.

  From this we conclude $\forall r.d \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap [\forall r]\varphi_n(\pi(d))$. Notice that $[\forall r]\varphi_n(\pi(d)) \subseteq M_{n+1}$ assured by Theorem 2 and the Definition of M_{n+1}. Applying Lemma 1 we get $\bigsqcap [\forall r]\varphi_n(\pi(d)) \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap ([\forall r]\varphi_n(\pi(d)))^{\mathfrak{B}_{n+1}}$ and by the first proposition of theorem 4 we have $\bigsqcap ([\forall r]\varphi_n(\pi(d)))^{\mathfrak{B}_{n+1}} \equiv_{\overrightarrow{\mathbb{K}}} \bigsqcap F_n^{-1}([\forall r]\varphi_n(\pi(d)))^{\mathfrak{B}_{n+1}} = \pi(c)$. So we have showed $c \equiv_{\overrightarrow{\mathbb{K}}} \pi(c)$. The application of F_n^{-1} assures $\pi(c) \in \mathcal{FLE}_n$.

- $c = \bigsqcap D$.
  W.l.o.g. we can assume, that every $d \in D$ has no conjunction outside the range of a quantifier, thus, one of the cases above is applicable. Therefore,

we know $\pi(d) \in \mathcal{FLE}_n$ and $d \equiv_{\mathbb{K}} \pi(d)$ for every $d \in D$. This implies $\bigsqcap D \equiv_{\mathbb{K}} \bigsqcap\{\pi(d) \mid d \in D\} = \pi(c)$ as well as $\pi(c) \in \mathcal{FLE}_n$.

□

In words, the π function just realizes the following transformation: beginning from "inside" the concept expression c, subformulae having maximal role depth of $n+1$ are substituted by equivalent ones with smaller role depth. When applied iteratively, this results in a formula $\tilde{c}$ from $\mathcal{FLE}_n$ that is equivalent to the original one. This formula's validity can now be checked by the method described in the preceding section.

It is easy to show, that the termination criterion mentioned above is equivalent to the finiteness of $\mathcal{FLE}/\equiv_{\mathbb{K}}$, which is trivially fulfilled if Δ is finite.

8 A Small Example

After having presented the algorithm in theory, we will consider an easy example for our method in order to show what type of information we can expect from this method. Let the universe Δ be the natural numbers including zero. Furthermore, let $M_\mathcal{C}$ and $M_\mathcal{R}$ be defined as shown in Figure 2 a). Carrying out the exploration on $\mathbb{K}_0$ (where the attributes M_0 are just the elements from $M_\mathcal{C}$ plus $\perp$) we get the implication base $\mathfrak{B}_0$ shown in Figure 2 b).

After this step, we generate the attribute set M_1 for the next one. First we reuse all attributes from M_0, second we take the conjunction over any $\mathfrak{B}_0$-closed subset not containing $\perp$ of M_0 preceded by an existential quantifier, and third we include all combinations of a universal quantifier and one attribute from M_0. Figure 3 lists the attributes from M_1.

$c \in M_\mathcal{C}$	name	$c^{I_\mathcal{C}}$
ev	even	$\{2n \mid n \in \mathbb{N}\}$
od	odd	$\{2n+1 \mid n \in \mathbb{N}\}$
pr	prime	$\{n \geq 2 \mid kl = n \Rightarrow k \in \{1, n\}\}$
e0	equals zero	$\{0\}$
e1	equals one	$\{1\}$
e2	equals two	$\{2\}$
g2	greater than two	$\{n \in \mathbb{N} \mid n \geq 3\}$

$r \in M_\mathcal{R}$	name	$r^{I_\mathcal{R}}$
s	successor	$\{(n, n+1) \mid n \in \mathbb{N}\}$
p	predecessor	$\{(n+1, n) \mid n \in \mathbb{N}\}$
d	divisor	$\{(m, n) \mid \exists k \in \mathbb{N} : m = kn\}$
m	multiple	$\{(n, m) \mid \exists k \in \mathbb{N} : m = kn\}$

$\{e0\}$	$\to \{ev\}$
$\{e1\}$	$\to \{od\}$
$\{e2\}$	$\to \{ev, pr\}$
$\{ev, pr\}$	$\to \{e2\}$
$\{od, pr\}$	$\to \{g2\}$
$\{pr, g2\}$	$\to \{od\}$
$\{ev, od\}$	$\to \{\perp\}$
$\{g2, e0\}$	$\to \{\perp\}$
$\{g2, e1\}$	$\to \{\perp\}$
$\{e0, e2\}$	$\to \{\perp\}$

Fig. 2. Attributes $M_\mathcal{C}$, $M_\mathcal{R}$ and definition of the incidence relations $I_\mathcal{C}$, $I_\mathcal{R}$ for the example and the implication base $\mathfrak{B}_0$ resulting from the first exploration step.

$g2$	pr	od	ev	$e1$	$e0$	$e2$	$\bot$			
$\exists s.\top$	$\exists s.g2$	$\exists s.pr$	$\exists s.od$	$\exists s.ev$	$\exists s.(od \sqcap g2)$	$\exists s.(od \sqcap e1)$	$\exists s.(ev \sqcap g2)$	$\exists s.(ev \sqcap e0)$	$\exists s.(od \sqcap g2 \sqcap pr)$	$\exists s.(ev \sqcap pr \sqcap e2)$
$\exists p.\top$	$\exists p.g2$	$\exists p.pr$	$\exists p.od$	$\exists p.ev$	$\exists p.(od \sqcap g2)$	$\exists p.(od \sqcap e1)$	$\exists p.(ev \sqcap g2)$	$\exists p.(ev \sqcap e0)$	$\exists p.(od \sqcap g2 \sqcap pr)$	$\exists p.(ev \sqcap pr \sqcap e2)$
$\exists m.\top$	$\exists m.g2$	$\exists m.pr$	$\exists m.od$	$\exists m.ev$	$\exists m.(od \sqcap g2)$	$\exists m.(od \sqcap e1)$	$\exists m.(ev \sqcap g2)$	$\exists m.(ev \sqcap e0)$	$\exists m.(od \sqcap g2 \sqcap pr)$	$\exists m.(ev \sqcap pr \sqcap e2)$
$\exists d.\top$	$\exists d.g2$	$\exists d.pr$	$\exists d.od$	$\exists d.ev$	$\exists d.(od \sqcap g2)$	$\exists d.(od \sqcap e1)$	$\exists d.(ev \sqcap g2)$	$\exists d.(ev \sqcap e0)$	$\exists d.(od \sqcap g2 \sqcap pr)$	$\exists d.(ev \sqcap pr \sqcap e2)$
$\forall s.g2$	$\forall s.pr$	$\forall s.od$	$\forall s.ev$	$\forall s.e1$	$\forall s.e0$	$\forall s.e2$	$\forall s.\bot$			
$\forall p.g2$	$\forall p.pr$	$\forall p.od$	$\forall p.ev$	$\forall p.e1$	$\forall p.e0$	$\forall p.e2$	$\forall p.\bot$			
$\forall m.g2$	$\forall m.pr$	$\forall m.od$	$\forall m.ev$	$\forall m.e1$	$\forall m.e0$	$\forall m.e2$	$\forall m.\bot$			
$\forall d.g2$	$\forall d.pr$	$\forall d.od$	$\forall d.ev$	$\forall d.e1$	$\forall d.e0$	$\forall d.e2$	$\forall d.\bot$			

Fig. 3. Attributes M_1 for the second exploration step.

Then we have to generate the a priori knowledge for the second exploration step. First, we use the information collected so far. When we proceed from the first $(i = 0)$ to the second $(i = 1)$ step we simply can use $\mathfrak{B}_0$ as additional a priori information without further adaption. Furthermore, applying the deduction consequences mentioned in Section 5 we have to add e.g.:

- $\{\bot\} \to M_1$
- $\{\exists s.(od \sqcap g2 \sqcap pr)\} \to \{\exists s.(od \sqcap g2)\}$
- $\{\exists s.pr, \forall s.g2\} \to \{\exists s.(od \sqcap g2 \sqcap pr)\}$
- $\{\forall p.ev, \forall p.od\} \to \{\forall p.e2\}$

After these preparations, the next exploration step is invoked. We visualize its result by the concept lattice in Figure 4, which mirrors the conceptual hierarchy of the formulae from M_1.

As an example, we will now demonstrate how to check the validity of the $\mathcal{FLE}_1$ entailment statement

$$pr \sqcap \exists s.(od \sqcap pr) \sqsubseteq_{\mathbb{K}} e2,$$

verbally: "is two the only prime having an odd prime sucessor?" Now, we carry out the necessary calculations and find

$$\begin{aligned}
&\varphi_1(pr \sqcap \exists s.(od \sqcap pr))\\
&= \varphi_1(pr) \cup \varphi_1(\exists s.(od \sqcap pr))\\
&= \varphi_1(pr) \cup [\exists r](\varphi_0(od \sqcap pr))^{\mathfrak{B}_0}\\
&= \varphi_1(pr) \cup [\exists r](\varphi_0(od) \cup \varphi_0(pr))^{\mathfrak{B}_0}\\
&= \{pr\} \cup [\exists r]\{od, pr\}^{\mathfrak{B}_0}\\
&= \{pr\} \cup [\exists r]\{od, pr, g2\}\\
&= \{pr, \exists r.(od \sqcap pr \sqcap g2)\}
\end{aligned}$$

as well as

$$\varphi_1(ev) = \{ev\}.$$

When applying the $\mathfrak{B}_1$-closure to both sets (the result is to large to be displayed here but can be derived from the line diagram next page) we find the outcomes

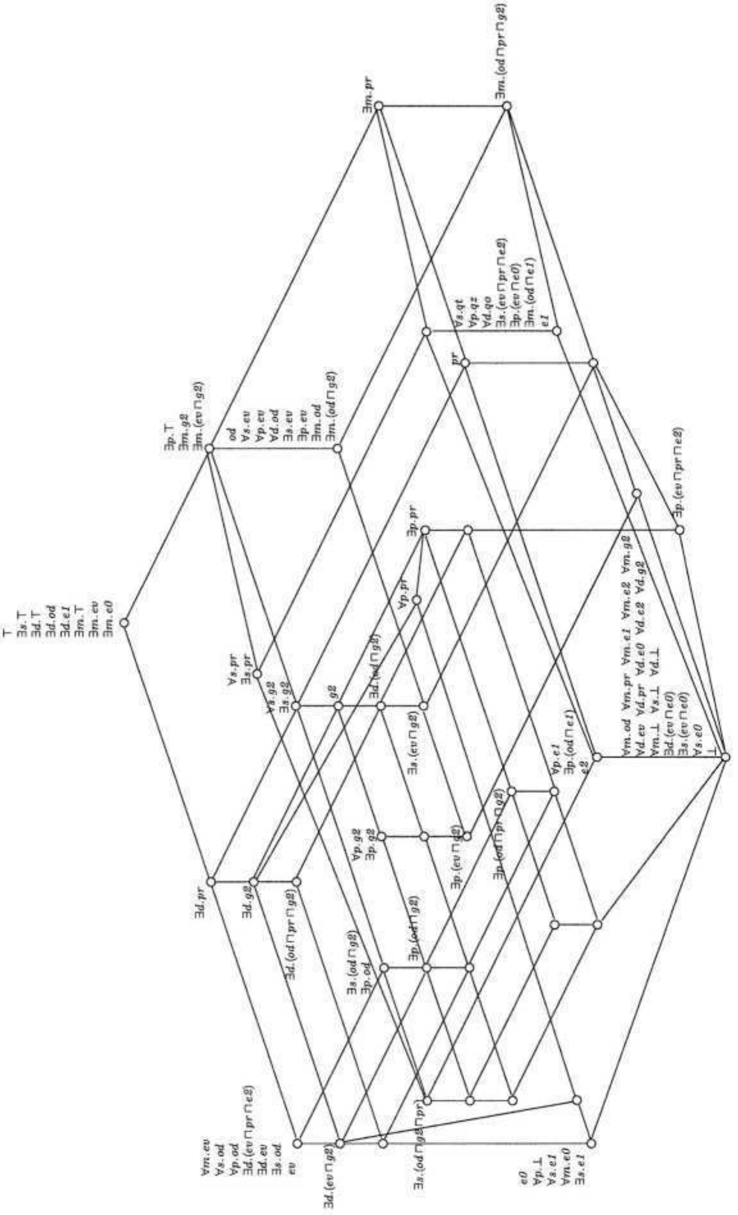


Fig. 4. Concept lattice from the second exploration step representing the implicational knowledge in $\mathbb{K}_1$.

even identical. Thus, the validity of our hypothetical entailment statement can be confirmed.

Finally we deal with the question whether the exploration algorithm terminates in our case after some step. This has to be denied for the following reason. Consider the infinite sequence $e0$, $\exists p.e0$, $\exists p.\exists p.e0, \ldots$. Every formula in this sequence is satisfied by exactly one natural number. Moreover, these numbers are all pairwise different. Therefore, every formula of the sequence is in another $\equiv_{\vec{\mathbb{K}}}$-equivalence class, thus $\mathcal{FLE}/\equiv_{\vec{\mathbb{K}}}$ is infinite in our case. Hence, the algorithm does not terminate.

9 A Possible Application: Ontology Exploration

After having dealt with details and theoretic properties of the algorithm as well as a small example we will now widen our scope and look for promising applications. As already said, we think the proposed algorithm could be very helpful in designing conceptual descriptions of world aspects. Markup ontologies are a very popular example for this. Although most of those descriptions are formulated in logics much more complex than $\mathcal{FLE}$ our algorithm is still applicable as long as there are complete reasoning algorithms for deciding subsumption and they contain $\mathcal{FLE}$ (for instance we have the FACT algorithm for reasoning in $\mathcal{SHIQ}(d)$ - see [10] and [9]). After stipulating the names and definitions of concepts and roles (and thereby specifying the fragment of reality to describe) the next step in designing an ontology would be to define axioms or rules stating how the specified concepts are interrelated. Our exploration algorithm can support this tedious and error-prone task by guiding the expert. Every potential axiom the algorithm comes up with will first be passed to the reasoning algorithm appropriate for the used logic. If this axiom can be proven it will be confirmed to the algorithm, if not the human expert has to be asked. If he judges the rule to be generally valid in the domain, a genuinely new axiom has been found and can be incorporated into the domain description. Otherwise the expert has to enter a counterexample, which violates the hypothetical axiom. One advantage of applying this technique is the guarantee, that all axioms expressable as $\mathcal{FLE}$ entailment statements with a certain role depth will certainly be found.

Finally we want to reply to a possible remark from the point of view of DL: one could object, that sometimes or even most times ontologies are designed for several different domains, such that an expert would not want to commit himself to one specific domain, as it is necessary when applying this algorithm. However, from the mathematical point of view this is not a severe problem: we just take the disjoint union of all domains we want to describe as reference domain of our exploration. A rule would be valid in this "superdomain" if and only if it is valid in all of the original domains.

For this reason we are very confident that an implementation of this algorithm could be a very helpful tool in order to build and refine domain descriptions - not only for working with ontologies. As there is a strong relationship between

DL and modal logic (which in turn can be enriched by temporal and epistemic features), applications for describing discrete dynamic systems and multi agent systems are in the realms of possibility.

References

1. F. Baader: Computing a Minimal Representation of the Subsumption Lattice of all Conjunctions of Concepts Defined in a Terminology. In: Proceedings of the International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency, KRUSE 95, Santa Cruz, USA, 1995.
2. Baader, F.: The Description Logic Handbook: Theory, Practice, and Applications. Cambridge University Press, 2003.
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, 2001.
4. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of prepositional Horn formulae. J. Logic Programming 3:267-284, 1984.
5. Ganter, B., Two basic algorithms in concept analysis. FB4-Preprint No 831, TH Darmstadt, 1984.
6. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin-Heidelberg, 1999.
7. Ganter, B., Rudolph, S., Formal Concept Analysis Methods for Dynamic Conceptual Graphs. In: H. S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base, Springer-Verlag, 2001.
8. Guigues, J.-L., Duquenne, V.: Families minimales d'implications informatives resultant d'un tableaux de donnés binaires. Math. Sci. Humaines 95, 1986.
9. Horrocks, I. et al.: The Ontology Inference Layer OIL,
URL: http://www.ontoknowledge.org/oil/papers.shtml.
10. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In: D. MacAllester (Ed.), Proceedings of CADE-2000, LNAI 1831, Springer, 2000.
11. Horrocks, I.: Benchmark analysis with fact. In: Proceedings of TABLEAUX 2000, LNAI 1847, Springer, 2000.
12. Prediger, S.: Terminologische Merkmalslogik in der Formalen Begriffsanalyse. In: G. Stumme, R. Wille (Eds.): Begriffliche Wissensverarbeitung: Methoden und Anwendungen. Springer-Verlag, Heidelberg, 2000.
13. Rudolph, S., An FCA Method for the Extensional Exploration of Relational Data. In: Aldo de Moor, Bernhard Ganter (Eds.): Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag, 2003.
14. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements, In: Artificial Intelligence, 48:1-26, 1991.
15. Sowa, J.: Ontology, Metadata, and Semiotics. In: B. Ganter / G. M. Mineau (Eds.): Conceptual Structures: Logical, Linguistic, and Computational Issues, LNAI 1867, Springer Verlag, 2000.
16. Wille, R.: Conceptual Graphs and Formal Concept Analysis. In: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (Eds.): Conceptual Structures: Fulfilling Peirce's Dream, Springer-Verlag, 1997.
17. Zickwolff, M.: Rule Exploration: First Order Logic in Formal Concept Analysis, PhD thesis, TH Darmstadt, 1991.

Appendix: A Deduction Calculus

Arising naturally from the considerations in this paper is the question for a deduction calculus for implications on $\mathcal{FLE}$. We found a set of deduction rules which we proved to be sound and complete. The completeness proof is based on a fixpoint construction of a standard model and beyond the scope and spatial capacity of this article. So we just present the deduction rules here.

Definition 7. *For given $M_\mathcal{C}, M_\mathcal{R}$ we define a ternary relation $Y \subseteq \mathcal{P}(\mathcal{FLE})^3$ as the smallest relation fulfilling the following conditions:*

$$\begin{aligned}
&(\emptyset, \{\bot\}, \emptyset) \in Y,\\
&(\Phi, \Phi_1, \Phi_2) \in Y \Rightarrow ([\exists r]\Phi, [\exists r]\Phi_1, [\exists r]\Phi_2) \in Y,\\
&(\Phi, \Phi_1, \Phi_2) \in Y \Rightarrow ([\forall r]\Phi, [\forall r]\Phi_1, [\exists r]\Phi_2) \in Y,\\
&(\Phi, \Phi_1, \Phi_2) \in Y \Rightarrow (\Psi \cup \Phi, \Psi \cup \Phi_1, \Psi \cup \Phi_2) \in Y,\\
&(\Phi, \Phi_1, \Phi_2) \in Y \Rightarrow (\Phi, \Phi_2, \Phi_1) \in Y.
\end{aligned}$$

The motivation for this definition is - roughly spoken - to encode case distinction. Note that, for all $(\Phi, \Phi_1, \Phi_2) \in Y$ and any binary power context family $\vec{\mathbb{K}}$, we have

$$\delta \in \llbracket \sqcap \Phi \rrbracket_{\vec{\mathbb{K}}} \Leftrightarrow \delta \in \llbracket \sqcap \Phi_1 \rrbracket_{\vec{\mathbb{K}}} \vee \delta \in \llbracket \sqcap \Phi_2 \rrbracket_{\vec{\mathbb{K}}}$$

for all $\delta \in \Delta$, verbally: every entity of the universe fulfilling all descriptions from Φ fulfills all descriptions from Φ_1 or all descriptions from Φ_2.

Definition 8. *The set of $\mathcal{DR}$* DERIVATION RULES *consists of the following rules $(a, b, c \in \mathcal{FLE},\ A, B_1, \ldots, B_n, C, D_1, \ldots, D_k \in \mathcal{P}_{fin}(\mathcal{FLE})$, and $(\Phi, \Phi_1, \Phi_2) \in Y)$*

$$\frac{}{\bot \to \{a\}}\mathcal{C} \qquad\qquad \frac{A \to B}{[\exists r]A \to [\exists r]B}\exists\mathcal{L}$$

$$\frac{}{\{\exists r.\bot\} \to \{\bot\}}\exists\mathcal{F} \qquad\qquad \frac{}{[\exists r]A \cup \{\forall r.b\} \to [\exists r](A \cup \{b\})}\forall\mathcal{P}$$

$$\frac{}{A \to A}\mathcal{ID} \qquad\qquad \frac{A \to B}{[\forall r]A \to [\forall r]B}\forall\mathcal{L}$$

$$\frac{A \to B}{A \cup \{c\} \to B}\mathcal{PE} \qquad\qquad \frac{\Phi_1 \to A, \Phi_2 \to A}{\Phi \to A}\mathcal{CD}$$

$$\frac{A \to B, A \cup B \to C}{A \to C}\mathcal{MP}$$

An Extension of the Theory of Information Flow to Semiconcept and Protoconcept Graphs

Grit Malik

Institute of Algebra
Department of Mathematics and Natural Sciences
Dresden University of Technology
Germany
malik@math.tu-dresden.de

Abstract. In this article the theory of distributed systems and information flow inside distributed systems by Barwise and Seligman is extended to semiconcept graphs and protoconcept graphs. For this purpose at first the information transferring mappings, *i.e.,* semiconcept graph morphisms and protoconcept graph morphisms are defined. As in the theory by Barwise and Seligman a minimal cover of a distributed system consisting of semiconcept or protoconcept graphs and the corresponding morphisms by a channel can be constructed. Such a channel consists of a set of graph morphisms into a common semiconcept or protoconcept graph which is called the core of the channel. The core represents the distributed system as a whole and in particular the common information of the semiconcept or protoconcept graphs in the distributed system.

1 Introduction

Semiconcept and protoconcept graphs are an extension of the theory of concept graphs. They include negation on the object level as well as on the relation level. Semiconcept graphs are at first semantically introduced by R. Wille in [16]. Their theory was further developed by J. Klinger in [4] and [5]. Especially syntax and semantics were separated. A further extension are protoconcept graphs. They include the theory of concept graphs, semiconcept graphs and some syntactical constructs of concept graphs with cuts. They were also introduced semantically by R. Wille in [17] and the separation of syntax and semantics is described by J. Klinger in [7]. For an overview see further [8].

Conceptual graphs are introduced by Sowa (*cf.* [14]) as a representation for logic. One of their application is knowledge representation. In particular in this area the treatment of negation is important. Joining the theory of semiconcept and protoconcept graphs with the theory of information flow will be promising for possible applications.

The theory of information flow by Barwise and Seligman considers distributed systems consisting of formal contexts[1] and infomorphisms between them (*cf.*

[1] Barwise and Seligman do not use the notion formal context

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 213–226, 2004.

[1]).It models how information flows inside such a distributed system how the information flow and the common information of the system can be considered as a whole.

In [10] a translation of the theory of distributed systems by J. Barwise and J. Seligman to distributed systems consisting of concept graphs is given. Concept graphs which are a mathematization of conceptual graphs were at first introduced by R. Wille in [15] and further developed by S. Prediger in [11] and [12]. It was shown that for a distributed system with concept graph a minimal cover by a channel can be constructed. Such a channel consists of a set of concept graph morphisms of the concept graphs of the system into a common concept graph. The common concept graph is supposed to represent the whole distributed system and in particular the common information of the concept graphs under the concept graph morphisms of the system. The channel morphisms are supposed to model the connection of the parts of the system to the whole system.

In this article at first distributed systems with semiconcept graphs are considered. But before starting to construct a cover of a distributed system with semiconcept graphs morphisms between semiconcept graphs are introduced analogously to concept graph morphisms. Here we have to reflect that semiconcept graphs include a kind of negation.

In the second part distributed systems with protoconcept graphs are considered. Again at first morphisms between protoconcept graphs are defined. Then a covering channel of a distributed system is constructed referring to the construction for semiconcept graphs.

2 Morphisms Between Semiconcept Graphs

In this section the definition of semiconcept graphs by J. Klinger (*cf.* [4] and [5]) is recalled and semiconcept graph morphisms are defined. At first alphabets for semiconcept graphs are introduced.

Definition 1. *An alphabet of semiconcept graphs is a triple $\mathcal{A} := (\mathcal{G}, \mathcal{S}, \mathcal{R})$ such that $\mathcal{G}$ is a finite set of object names, $(\mathcal{S}, \leq_{\mathcal{S}})$ is a finite ordered set of semiconcept names and $(\mathcal{R}, \leq_{\mathcal{R}})$ is a finite ordered set of relation names which is partitioned into the ordered sets $(\mathcal{R}_k, \leq_{\mathcal{R}_k})$, $k = 1, \ldots, n$. $\mathcal{S}$ has a greatest element $\top_{\mathcal{S}}$ and a smallest element $\bot_{\mathcal{S}}$. Furthermore the sets $\mathcal{R}_k$, $k \in \mathbb{N}$, have a greatest element $\top_{\mathcal{R}_k}$ and a smallest element $\bot_{\mathcal{R}_k}$.*

The underlying structure of a semiconcept graph is a relational graph.

Definition 2. *A relational graph is a triple (V, E, ν) such that*

- *V is a set of vertices,*
- *E is a set of edges, and*
- *$\nu : E \to \bigcup_{k \in \mathbb{N}} V^k$ is a mapping.*

The function ν assigns to every edge a tuple of vertices. If $\nu(e) = (v_1, \ldots, v_k)$ then$|e| := k$ is called the arity of the edge e. If V and E are finite the relational *graph is called finite. We set* $E^{(k)} := \{e \in E \mid |e| = k\}$ *for all* $k \in \mathbb{N}$. *Instead of* $\nu(e)|_i$ *often* $e|_i$ *is written.*

A semiconcept graph is a relational graph with labeled edges and vertices. The edges are labeled by relation names and the vertices are labeled by semiconcept names. Further to every vertex a positive and a negative reference is assigned.

Definition 3. *A semiconcept graph over an alphabet* $\mathcal{A} = (\mathcal{G}, \mathcal{S}, \mathcal{R})$ *is a tuple* $(V, E, \nu, \kappa, \rho)$ *such that*

- (V, E, ν) *is a relational graph,*
- $\kappa : V \cup E \to \mathcal{S} \cup \mathcal{R}$ *is a function such that* $\kappa(V) \subseteq \mathcal{S}$ *and* $\kappa(E^{(k)}) \subseteq \mathcal{R}_k$, *and*
- $\rho : V \to \mathfrak{P}(G) \times \mathfrak{P}(G)$ *is a function with* $v \mapsto (\rho^+(v), \rho^-(v))$ *whereas* $\rho^+ : V \to \mathfrak{P}(G)$ *and* $\rho^- : V \to \mathfrak{P}(G)$ *are mappings such that*
  1. $\rho^+(v) \cup \rho^-(v) \neq \emptyset$ *for all* $v \in V$,
  2. $\rho^+(e) \cup \rho^-(e) \neq \emptyset$ *for all* $e \in E$,
  3. $\kappa(v) \geq \kappa(w)$ *implies* $\rho^-(v) \cap \rho^+(w) = \emptyset$ *for all* $v, w \in V$, *and*
  4. $\kappa(e) \geq \kappa(f)$ *implies* $\rho^-(e) \cap \rho^+(f) = \emptyset$ *for all* $e, f \in E^{(k)}$.

  Here $\rho^+(e)$ *denotes* $\rho^+(v_1) \times \cdots \times \rho^+(v_k)$ *and* $\rho^-(e) = \rho^-(v_1) \times \cdots \times \rho^-(v_k)$ *for every edge* e *with* $\nu(e) = (v_1, \ldots, v_k)$.

Figure 1 shows an example of three semiconcept graphs. They represent properties of some chemical elements (*cf.* [13]). The interpretation of the first graph is that the alkali metal potassium is weaker than the alkali metal sodium. Further magnesium and calcium are no alkali metals and magnesium is not weaker than calcium. The semantics of semiconcept graphs is interpreted in power context families (see [4]).

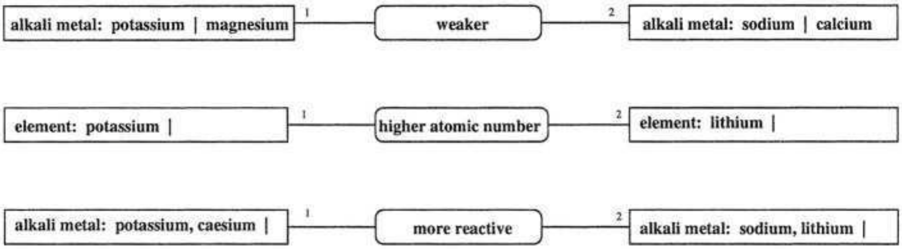

alkali metal: potassium | magnesium
1
weaker
2
alkali metal: sodium | calcium
element: potassium |
1
higher atomic number
2
element: lithium |
alkali metal: potassium, caesium |
1
more reactive
2
alkali metal: sodium, lithium |

Fig. 1. Semiconcept graphs representing properties of chemical elements

In [9] at first concept graph morphisms were introduced. It turned out that the slightly changed version in [10] is more useful to model the information flow. They correspond to the infomorphisms between formal contexts which were introduced by J. Barwise and J. Seligman in [1]. Infomorphisms are contravariant pairs of functions satisfying a special condition. They are supposed to model the information flow between the contexts. The direction determining function goes between the attributes of the contexts. The second function goes between the objects into the other direction such that an object is incident to the image of an attribute in the second context iff the image of the object is incident to the attribute in the first context. For an introduction to Formal Concept Analysis see [2]. The different directions of the functions were introduced to formalize that information always flows in both directions.

Since the underlying structure of a concept graph and a semiconcept graph is a relation graph the underlying structure of concept graph morphisms and semiconcept graph morphisms are relational graph morphisms. Analogously to infomorphisms they consist of two contravariant functions. The direction determining function goes between the edges. The second function between the vertices goes into the other direction. The condition says that a vertex v of the second relational graph is adjacent to the image of an edge $f_E(e)$ iff the image of the vertex $f_V(v)$ is adjacent to the edge e in the first relational graph.

Definition 4. *A relational graph morphism* $f : (V_1, E_1, \nu_1) \rightleftarrows (V_2, E_2, , \nu_2)$ *is a contravariant pair* (f_E, f_V) *of functions* $f_E : E_1 \to E_2$ *and* $f_V : V_2 \to V_1$ *in the other direction such that for all* $e \in E_1$ *and all* $v \in V_2$

$$\exists_{i \leq |e|} f_V(v) = \nu_1(e)|_i \iff \exists_{j \leq |f_E(e)|} v = \nu_2(f_E(e))|_j$$

is satisfied.

Now semiconcept graph morphisms can be defined as relational graph morphisms with three additional functions between the alphabets of the semiconcept graphs. These three functions are compatible with the vertex and the edge function. Conditions 2 and 3 say that images of labels of v and e are labels of the images of $f_V(v)$ and $f_E(e)$, respectively. In the condition 4 and 5 it is required that the positive or negative references of the image of a vertex contain the image of its set of positive and negative references, respectively. Hence semiconcept graph morphisms are relational graph morphisms the direction determining function is the function between the edges.

Definition 5. *Let* $\mathfrak{G}_1 := (V_1, E_1, \nu_1, \kappa_1, \rho_1)$ *be a semiconcept graph over the alphabet* $(\mathcal{G}_1, \mathcal{S}_1, \mathcal{R}_1)$ *and* $\mathfrak{G}_1 := (V_2, E_2, \nu_2, \kappa_2, \rho_2)$ *a semiconcept graph over* $(\mathcal{G}_2, \mathcal{S}_2, \mathcal{R}_2)$. *A semiconcept graph morphism* $f := (f_E, f_V, f_C, f_S, f_{\mathcal{R}}) : \mathfrak{G}_1 \rightleftarrows \mathfrak{G}_2$ *consists of five functions*

$$f_E : E_1 \to E_2, \quad f_V : V_2 \to V_1, \quad f_{\mathcal{G}} : \mathcal{G}_2, \to \mathcal{G}_1 \quad f_S : \mathcal{S}_2 \to \mathcal{S}_1, \quad f_{\mathcal{R}} : \mathcal{R}_1 \to \mathcal{R}_2$$

such that

1. (f_E, f_V) *is a relational graph morphism,*
2. $f_S(\kappa_2(v)) = \kappa_1(f_V(v))$ *for all* $v \in V_2$
3. $f_{\mathcal{R}}(\kappa_1(e)) = \kappa_2(f_E(e))$ *for all* $e \in E_1$
4. $\rho_1^+(f_V(v)) \supseteq f_{\mathcal{G}}(\rho_2^+(v)$ *for all* $v \in V_2$
5. $\rho_1^-(f_V(v)) \supseteq f_{\mathcal{G}}(\rho_2^-(v))$ *for all* $v \in V_2$

The identity function $id := (id_E, id_V, id_\mathcal{G}, id_\mathcal{S}, id_\mathcal{R})$ is a semiconcept graph morphism from a semiconcept graph to itself. Further the composition of two semiconcept graph morphisms $f : \mathfrak{G}_1 \rightleftarrows \mathfrak{G}_2$ and $g : \mathfrak{G}_2 \rightleftarrows \mathfrak{G}_3$ is again a semiconcept graph morphism and is defined by $g \circ f := (g_E \circ f_E, f_V \circ g_V, f_\mathcal{G} \circ g_\mathcal{G}, f_\mathcal{S} \circ g_\mathcal{S}, g_\mathcal{R} \circ f_\mathcal{R})$.

In Figure 2 an example of two semiconcept graph morphisms between the semiconcept graphs of Figure 1 is given. These two semiconcept graph morphisms are just two examples of more possible morphisms. The decision which semiconcept graph morphisms is chosen depends on the user and on the facts represented in the semiconcept graphs. Of course it is possible to find semiconcept graph morphisms which map for instance vertices representing unrelated objects in real world. But they would never be chosen if they are not in the users interest.

3 Covering Distributed Systems with Semiconcept Graphs

In [1] distributed systems consisting of formal contexts and infomorphisms are considered. The step to distributed systems with concept graphs was done in [10]. Now distributed systems consisting of a set of semiconcept graphs and a set of semiconcept graph morphisms between them are considered and a minimal cover by a channel is constructed. This construction is analogous to the construction of a minimal cover of distributed systems with concept graphs which is a translation of the construction by Barwise and Seligman for distributed systems with formal contexts and infomorphisms. But here we have a few restrictions to teh distributed system.

Definition 6. *A distributed system* $\mathfrak{D} := (\mathrm{scg}(\mathfrak{D}), \mathrm{scgm}(\mathfrak{D}))$ *consists of a finite set* $\mathrm{scg}(\mathfrak{D})$ *of semiconcept graphs and a set* $\mathrm{scgm}(\mathfrak{D})$ *of semiconcept graph morphisms between semiconcept graphs of* $\mathrm{scg}(\mathfrak{D})$, *i.e.,* $\mathrm{scgm}(\mathfrak{D}) \subseteq \{f : \mathfrak{G}_1 \rightleftarrows \mathfrak{G}_2 \mid \mathfrak{G}_1, \mathfrak{G}_2 \in \mathrm{scg}(\mathfrak{D})\}$.

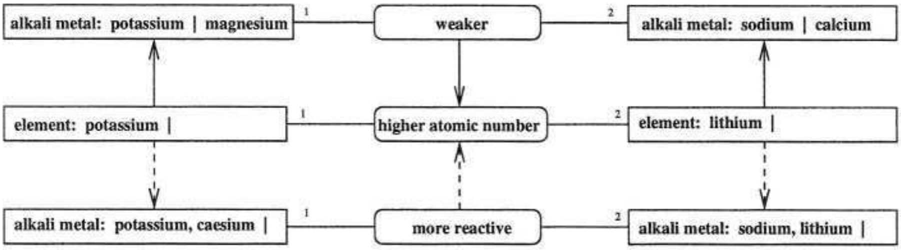
It is possible that a semiconcept graph occurs more than once in the distributed system, *i.e.,* $\mathfrak{G}_i = \mathfrak{G}_j$ for $i \neq j$.

Definition 7. *A channel C is a set* $\{\gamma_k : \mathfrak{G}_k \rightleftarrows \mathfrak{C} \mid k = 1, \dots, n\}$ *of semiconcept graph morphisms into a common semiconcept graph* $\mathfrak{C}$. $\mathfrak{C}$ *is called the core of C.*

Channels are a possibility to represent a distributed system as a whole object. The morphisms of the channel are supposed to represent the connection between the parts of the distributed system to the whole system. The next definition gives the connection between distributed systems and channels.

Definition 8. *A channel* $C := \{\gamma_k : \mathfrak{G}_k \rightleftarrows \mathfrak{C} \mid k = 1, \dots, n\}$ *covers a distributed system* $\mathfrak{D}$ *with* $\mathrm{scg}(\mathfrak{D}) = \{\mathfrak{G}_k \mid k = 1, \dots n\}$ *if for every* $i, j \leq n$ *and every semiconcept graph morphism* $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j \in \mathrm{scgm}(\mathfrak{D})$ *holds*

$$\gamma_i = \gamma_j \circ f.$$


alkali metal: potassium | magnesium
1
weaker
2
alkali metal: sodium | calcium
element: potassium |
1
higher atomic number
2
element: lithium |
alkali metal: potassium, caesium |
1
more reactive
2
alkali metal: sodium, lithium |

$f : \mathfrak{G}_1 \rightleftarrows \mathfrak{G}_2$
$f_E : E_1 \to E_2$
(weaker) $\mapsto$ (higher atomic number)
$f_V : V_2 \to V_1$
[element: potassium |] $\mapsto$ [alkali metal: potassium | magnesium]
[element: lithium |] $\mapsto$ [alkali metal: sodium | calcium]
$f_{\mathcal{G}} : \mathcal{G}_2 \to \mathcal{G}_1$
potassium $\mapsto$ potassium
lithium $\mapsto$ sodium
$f_{\mathcal{S}} : \mathcal{S}_2 \to \mathcal{S}_1$
element $\mapsto$ alkali metal
$f_{\mathcal{R}} : \mathcal{R}_1 \to \mathcal{R}_2$
weaker $\mapsto$ higher atomic number

$g : \mathfrak{G}_2 \rightleftarrows \mathfrak{G}_2$
$g_E : E_3 \to E_2$
(more reactive) $\mapsto$ (higher atomic number)
$g_V : V_2 \to V_3$
[element: potassium |] $\mapsto$ [alkali metal: potassium, caesium |]
[element: lithium |] $\mapsto$ [alkali metal: sodium, lithium |]
$g_{\mathcal{G}} : \mathcal{G}_2 \to \mathcal{G}_3$
potassium $\mapsto$ potassium
lithium $\mapsto$ lithium
$g_{\mathcal{S}} : \mathcal{S}_2 \to \mathcal{S}_3$
element $\mapsto$ alkali metal
$g_{\mathcal{R}} : \mathcal{R}_3 \to \mathcal{R}_2$
more reactive $\mapsto$ higher atomic number

Fig. 2. Examples for morphisms between semiconcept graphs

C is a minimal cover of $\mathfrak{D}$ if for every other channel C' covering $\mathfrak{D}$ there is a unique semiconcept graph morphism from $\mathfrak{C}$ to $\mathfrak{C}'$.

As for concept graphs and concept graph morphisms such a minimal cover by a channel can be constructed if the distributed system satisfies some conditions. The construction can be done stepwise by constructing at first the semiproduct of the semiconcept graphs and afterwards a quotient by an invariant. Those operations are similar to the analogous operations for concept graphs. But because of the negations some technical difficulties have to be solved in the constructions. In particular we have to restrict to distributed systems satisfying the following conditions:

1. every semiconcept graph is image or preimage of a semiconcept graph morphism,
2. the edge function of a semiconcept graph morphism preserve the arities of the edges, *i.e.*, $|f_E(e)| = |e|$, and
3. the vertex function is injective.

Otherwise we are not able to construct the core of the channel. Here the channel is constructed directly. For comparison in the theory by Barwise and Seligman the core of the channel has the Cartesian product of the object sets of the formal contexts as objects and the disjoint union of the attribute sets as attributes.

Construction. Let $\mathfrak{D}$ be a distributed system with $\mathrm{scg}(\mathfrak{D}) = \{\mathfrak{G}_k \mid k = 1,\ldots,n\}$ satisfying the conditions 1-3. The limit $\lim(\mathfrak{D})$ of $\mathfrak{D}$ is the channel constructed as follows.

1. The core $\mathfrak{C}$ of $\lim(\mathfrak{D})$:
   - The alphabet $\mathcal{A}_{\mathfrak{C}}$ of the core has the Cartesian product $\mathcal{G}_1 \times \ldots \times \mathcal{G}_n$ of the object names of the semiconcept graphs $\mathfrak{G}_i$ as the set of object names $\mathcal{G}_{\mathfrak{G}}$. The set $\mathcal{S}_{\mathfrak{C}}$ of semiconcept names is $\mathcal{S}_1 \times \ldots \times \mathcal{S}_n$. The set of relation names $\mathcal{R}_{\mathfrak{C}}$ is obtained by factorizing the disjoint union $\dot{\bigcup}_{k=1}^{n} \mathcal{R}_k$ by the smallest equivalence relation containing the relation $\eta_{\mathcal{R}}$ which is defined by $(r_i, r_j) \in \eta_{\mathcal{R}}$, $r_i \in \mathcal{R}_i$ and $r_j \in \mathcal{R}_j$, if there is a semiconcept graph morphism $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j$ with $f_{\mathcal{R}}(r_i) = r_j$. The order is defined by $[r] \leq [s] \iff \forall_{x \in [r]} \exists_{y \in [s]} x \leq y$.
   - The set of vertices $V_{\mathfrak{C}}$ is the set of all tuples $(v_1, \ldots, v_n)$ of vertices in $V_1 \times \ldots \times V_n$ such that
     - for all semiconcept graph morphisms $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j \in \mathrm{scgm}(\mathfrak{D})$ $f(v_j) = v_i$ hold and
     - $\rho_1^+(v_1) \times \ldots \times \rho_n^+(v_n) \cup \rho_1^-(v_1) \times \ldots \times \rho_n^-(v_n) \neq \emptyset$ holds.
   - The set of edges $E_{\mathfrak{C}}$ is analogous to the relation names the disjoint union $\dot{\bigcup}_{k=1}^{n} E_k$ factorized by the smallest equivalence relation containing the relation η_E. η_E is defined by $(e_i, e_j) \in \eta_E$, $e_i \in E_i$ and $e_j \in E_j$, iff there is a semiconcept graph morphisms $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j \in \mathrm{scgm}(\mathfrak{D})$ with $f_E(e_i) = e_j$.
   - A tuple of vertices $(v_1, \ldots, v_n)$ is adjacent to an edge $[e_i]$ iff v_i is adjacent to e_i in $\mathfrak{G}_i$. Because of the definition of η_E, the properties of semiconcept graph morphisms, and the restrictions of $\mathfrak{D}$ the function ν is well-defined. The indices of the adjacent vertices $(v_1, \ldots, v_n)$ of an edge $[e]$ are ordered at first by the minimality of the indices which the vertices have for the edges in $[e]$, then by the frequency of occurrence of the minimal number, and at last arbitrarily.

- $\kappa_{\mathfrak{C}} : V_{\mathfrak{C}} \cup E_{\mathfrak{C}} \to \mathcal{S}_{\mathfrak{C}} \cup \mathcal{R}_{\mathfrak{C}}$ is defined by

$$\kappa_{\mathfrak{C}}(v_1, \ldots, v_n) = (\kappa_1(v_1), \ldots, \kappa_n(v_n))$$
$$\kappa_{\mathfrak{C}}([e_i]) = [\kappa_i(e_i)].$$

$\kappa_{\mathfrak{C}}$ is again well-defined because of the definition of η_E and $\eta_{\mathcal{R}}$ and of the properties of semiconcept graph morphisms.

- $\rho^+_{\mathfrak{C}} : V_{\mathfrak{C}} \to \mathfrak{P}(\mathcal{G}_{\mathfrak{C}})$ is defined by $\rho^+_{\mathfrak{C}}(v_1, \ldots, v_n) = \rho^+_1(v_1) \times \ldots \times \rho^+_n(v_n)$ and $\rho^-_{\mathfrak{C}} : V_{\mathfrak{C}} \to \mathfrak{P}(\mathcal{G}_{\mathfrak{C}})$ is defined by $\rho^-_{\mathfrak{C}}(v_1, \ldots, v_n) = \rho^-_1(v_1) \times \ldots \times \rho^-_n(v_n)$.

2. The semiconcept graph morphisms $\gamma^i : \mathfrak{G}_i \rightleftarrows \mathfrak{C}$:
   - $\gamma^i_E : E_i \to E_{\mathfrak{C}}$ is defined by $\gamma^i_E(e) = [e]$.
   - $\gamma^i_V : V_{\mathfrak{C}} \to V_i$ is defined by $\gamma^i_V(v_1, \ldots, v_n) = v_i$.
   - $\gamma^i_{\mathcal{G}} : \mathcal{G}_{\mathfrak{C}} \to \mathcal{G}_i$ is defined by $\gamma^i_{\mathcal{G}}(g_1, \ldots, g_n) = g_i$.
   - $\gamma^i_{\mathcal{S}} : \mathcal{S}_{\mathfrak{C}} \to \mathcal{S}_i$ is defined by $\gamma^i_{\mathcal{S}}(s_1, \ldots, s_n) = s_i$.
   - $\gamma^i_{\mathcal{R}} : \mathcal{R}_i \to \mathcal{R}_{\mathfrak{C}}$ is defined by $\gamma^i_{\mathcal{R}}(r) = [r]$.

Theorem 1. *Let $\mathfrak{D}$ be a distributed system satisfying the condition 1-3. Then* $\lim(\mathfrak{D})$ *is a minimal cover and it is unique up to isomorphism.*

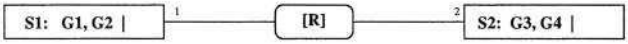
The result of Theorem 1 is proved similar to the analogous result in [10] for distributed systems with concept graphs. But the treatment of the negation leads to some more technical difficulties. The restrictions to the distributed system are necessary to obtain a sound semiconcept graph as core. Thus not every arbitrary distributed system with semiconcept graphs can be covered.

Figure 3 shows the core of the distributed system consisting of the three semiconcept graphs in Figure 1 and the semiconcept graph morphisms in Figure 2. It represents the common information of the distributed system. This information is that chemical elements (or at least the elements considered here) of the first main groups are weaker and more reactive with higher atomic number. The semantics can be interpreted in a standard power context family. For practical applications probably an isomorphic alphabet is chosen.

4 Morphisms Between Protoconcept Graphs

In this section at first the syntax of protoconcept graphs is given which was introduced by J. Klinger in [7]. See further [8]. It differs only a little of the syntax of semiconcept graphs. In the set of relation names a special element $\doteq$ is added.

Definition 9. *An alphabet of protoconcept graphs is a triple $\mathcal{A} := (\mathcal{G}, \mathcal{P}, \mathcal{R})$ such that $\mathcal{G}$ is a finite set of object names, $(\mathcal{P}, \leq_{\mathcal{P}})$ is a finite ordered set of protoconcept names and $(\mathcal{R}, \leq_{\mathcal{R}})$ is a finite ordered set of relation names which is partitioned into the ordered sets $(\mathcal{R}_k, \leq_{\mathcal{R}_k})$, $k = 1, \ldots, n$.*
$\mathcal{P}$ has a greatest element $\top_{\mathcal{P}}$ and a smallest element $\bot_{\mathcal{P}}$. Again the sets $\mathcal{R}_k$, $k \in \mathbb{N}$, have a greatest element $\top_{\mathcal{R}_k}$ and a smallest element $\bot_{\mathcal{R}_k}$, too. Further the set $\mathcal{R}_2$ is equipped with a special element $\doteq$ which is called identity.



semiconcept names	
S1 = S2:	(alkali metal, element, alkali metal)
object names	
G1:	(potassium, potassium, potassium)
G2:	(potassium, potassium, caesium)
G3:	(sodium, lithium, sodium)
G4:	(sodium, lithium, lithium)
relation names	
[R]:	{weaker, higher atomic number, more reactive}

Fig. 3. The core of the channel

Now the definition of a quasi protoconcept graph is recalled. The conditions are exactly the same conditions as for semiconcept graphs.

Definition 10. *A quasi protoconcept graph over an alphabet $\mathcal{A} = (\mathcal{G}, \mathcal{P}, \mathcal{R})$ is a tuple $(V, E, \nu, \kappa, \rho)$ such that*

- *(V, E, ν) is a relational graph,*
- *$\kappa : V \cup E \to \mathcal{P} \cup \mathcal{R}$ a function such that $\kappa(V) \subseteq \mathcal{P}$ and $\kappa(E^{(k)}) \subseteq \mathcal{R}_k$, and*
- *$\rho : V \to \mathfrak{P}(G) \times \mathfrak{P}(G)$ is a function with $v \mapsto (\rho^+(v), \rho^-(v))$ whereas $\rho^+ : V \to \mathfrak{P}(G)$ and $\rho^- : V \to \mathfrak{P}(G)$ are mappings such that*
  1. *$\rho^+(v) \cup \rho^-(v) \neq \emptyset$ for all $v \in V$,*
  2. *$\rho^+(e) \cup \rho^-(e) \neq \emptyset$ for all $e \in E$,*
  3. *$\kappa(v) \geq \kappa(w)$ implies $\rho^-(v) \cap \rho^+(w) = \emptyset$ for all $v, w \in V$, and*
  4. *$\kappa(e) \geq \kappa(f)$ implies $\rho^-(e) \cap \rho^+(f) = \emptyset$ for all $e, f \in E^{(k)}$.*

  Here $\rho^+(e)$ denotes $\rho^+(v_1) \times \cdots \times \rho^+(v_k)$ and $\rho^-(e) = \rho^-(v_1) \times \cdots \times \rho^-(v_k)$ for every edge e with $\nu(e) = (v_1, \ldots, v_k)$.

A quasi protoconcept graph $\mathfrak{G}$ is *satisfiable* if there is a model $(\vec{\mathbb{K}}, \lambda)$ for $\mathfrak{G}$. Such a model consists of a power context family $\vec{\mathbb{K}}$ and an interpretation λ of the alphabet $\mathcal{A}$ in $\vec{\mathbb{K}}$. A power context family $\vec{\mathbb{K}}$ is a tuple $(\mathbb{K}_0, \mathbb{K}_1, \ldots, \mathbb{K}_n)$ of formal contexts such that $G_k \subseteq G_0^k$. The interpretation λ maps object names to objects of $\mathbb{K}_0$, protoconcept names to protoconcepts of $\mathbb{K}_0$ and k-ary relation names to protoconcepts of $\mathbb{K}_k$ such that an edge and a vertex condition are satisfied. For more details see [7].

We will now consider only satisfiable quasi protoconcept graphs. Figure 4 shows an example of two protoconcept graphs.

Definition 11. *A protoconcept graph is a satisfiable quasi protoconcept graph.*

Now it is possible to define protoconcept graph morphisms as the analogous to semiconcept graph morphisms and concept graph morphisms. The definition slightly differs from the definition of semiconcept graph morphisms because we have two additional conditions for treating the relation name $\doteq$.

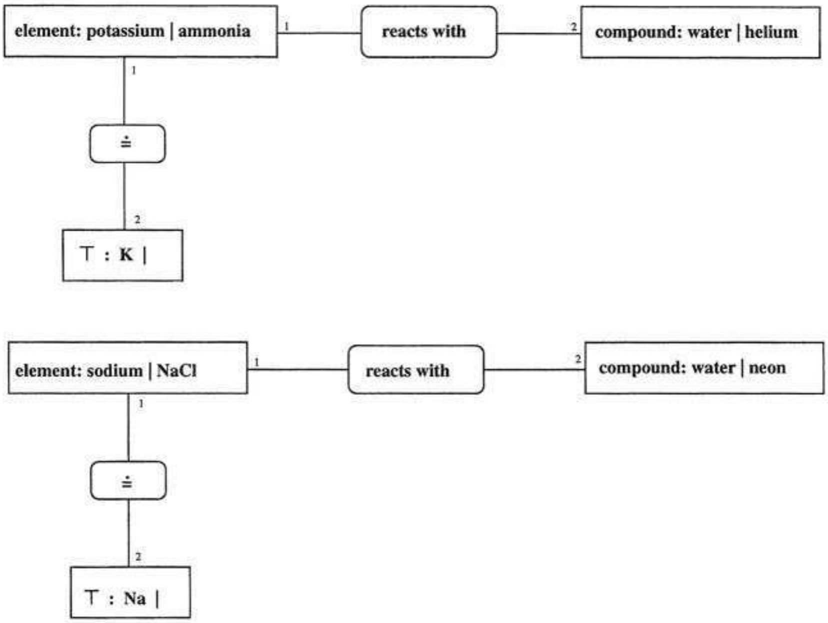


Fig. 4. Protoconcept graphs representing properties of chemical substances

Definition 12. *Let $\mathfrak{G}_1 := (V_1, E_1, \nu_1, \kappa_1, \rho_1)$ be a protoconcept graph over the alphabet $(\mathcal{G}_1, \mathcal{P}_1, \mathcal{R}_1)$ and $\mathfrak{G}_2 := (V_2, E_2, \nu_2, \kappa_2, \rho_2)$ a protoconcept graph over $(\mathcal{G}_2, \mathcal{P}_2, \mathcal{R}_2)$. A protoconcept graph morphism $f := (f_E, f_V, f_{\mathcal{C}}, f_{\mathcal{P}}, f_{\mathcal{R}}) : \mathfrak{G}_1 \rightleftarrows \mathfrak{G}_2$ consists of five functions*

$$f_E : E_1 \to E_2, \quad f_V : V_2 \to V_1, \quad f_{\mathcal{G}} : \mathcal{G}_2 \to \mathcal{G}_1, \quad f_{\mathcal{P}} : \mathcal{P}_2 \to \mathcal{P}_1, \quad f_{\mathcal{R}} : \mathcal{R}_1 \to \mathcal{R}_2$$

such that

1. *(f_E, f_V) is a relational graph morphism,*
2. *$f_{\mathcal{P}}(\kappa_2(v)) = \kappa_1(f_V(v))$ for all $v \in V_2$*
3. *$f_{\mathcal{R}}(\kappa_1(e)) = \kappa_2(f_E(e))$ for all $e \in E_1$*
4. *$f_{\mathcal{R}}(\doteq_1) = \doteq_2$*
5. *$f_{\mathcal{R}}(r) \neq \doteq_2$ for all $r \in \mathcal{R}_1, r \neq \doteq$*
6. *$\rho_1^+(f_V(v)) \supseteq f_{\mathcal{G}}(\rho_2^+(v)$ for all $v \in V_2$*
7. *$\rho_1^-(f_V(v)) \supseteq f_{\mathcal{G}}(\rho_2^-(v))$ for all $v \in V_2$*

In Figure 5 a possible protoconcept graph morphism between the protoconcept graphs in Figure 4 is pictured.

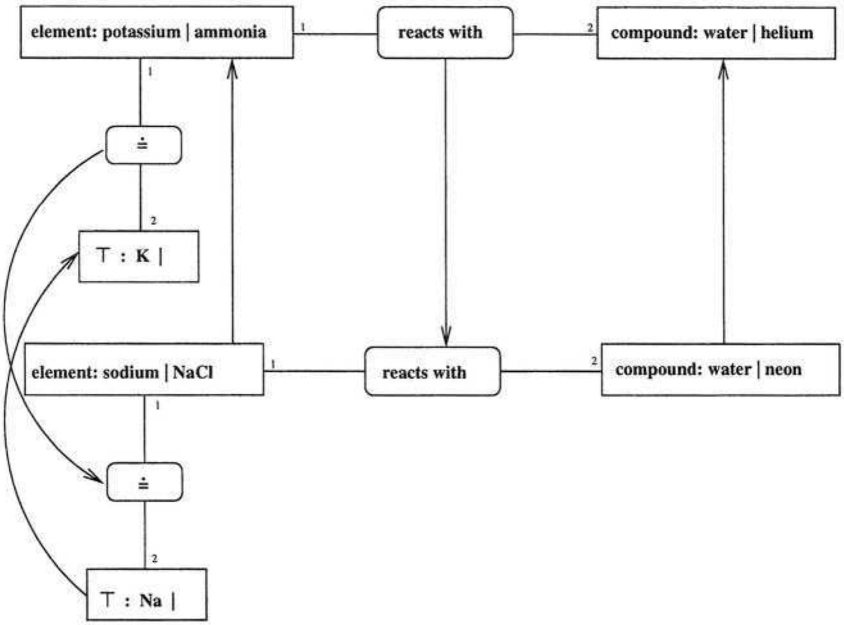


Fig. 5. An example of a protoconcept graph morphism

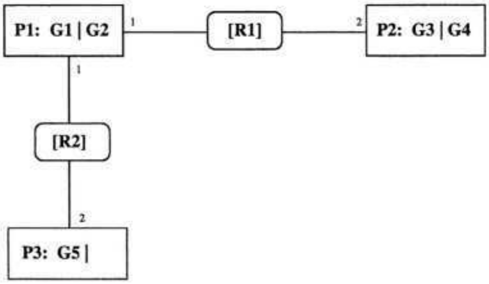
5 Covering Distributed Systems with Protoconcept Graphs

After defining morphisms between protoconcept graphs we can again consider distributed systems $\mathfrak{D}$ consisting now of a finite set $\mathrm{pcg}(\mathfrak{D}) := \{\mathfrak{G}_i \mid i = 1, \ldots, n\}$ of protoconcept graphs and a set $\mathrm{pcgm}(\mathfrak{G})$ of protoconcept graph morphisms between the protoconcept graphs of $\mathrm{pcg}(\mathfrak{D})$. The definitions of channels and minimal covers are exactly the definitions in section 3 only with 'semiconcept graph' replaced by 'protoconcept graph' and 'semiconcept graph morphism' replaced by 'protoconcept graph morphism'.

Again distributed system with protoconcept graphs are restricted to satisfy the conditions

1. every protoconcept graph is image or preimage of a protoconcept graph morphism,
2. the edge function of a protoconcept graph morphism preserve the arities of the edges, *i.e.*, $|f_E(e)| = |e|$, and
3. the vertex function is injective.

Then they can be covered by a channel consisting of a set of protoconcept graph morphisms from the protoconcept graphs of $\mathrm{pcg}(\mathfrak{D})$ into a common protoconcept graph, the core of the channel.



semiconcept names		object names	
P1:	(element, element)	G1:	(potassium, sodium)
P2:	(compound, compound)	G1:	(potassium, sodium)
P3:	$(\top, \top)$	G3:	(water, water)
relation names		G4:	(helium, neon)
[R1]:	{(reacts with,1), (reacts with,2)}	G5:	(K, Na)
[R1]:	{ $(\doteq,1)$, $(\doteq,2)$ }		

Fig. 6. The core of a covering channel

Construction. Let $\mathfrak{D}$ be a distributed system with $\mathrm{pcg}(\mathfrak{D}) := \{\mathfrak{G}_i \mid i = 1, \ldots, n\}$. The limit $\lim(\mathfrak{D})$ is the channel constructed as follows.

1. At first the core $\mathfrak{C}$ of $\lim \mathfrak{D}$ is constructed.
   - The alphabet $\mathcal{A}_{\mathfrak{C}} = (\mathcal{G}_{\mathfrak{C}}, \mathcal{P}_{\mathfrak{C}}, \mathcal{R}_{\mathfrak{C}})$ is constructed analogously to the alphabet in the case of semiconcept graphs. The sets of semiconcept names are here replaced by the sets of the protoconcept names of the protoconcept graphs in $\mathrm{pcg}(\mathfrak{D})$.
     It is not necessary to add a new $\doteq$ - sign in $\mathcal{R}_{\mathfrak{C}}$. This depends on the conditions 4 and 5 of the definition of protoconcept graph morphisms.
   - The set of vertices $V_{\mathfrak{C}}$ is the set of all tuples $(v_1, \ldots, v_n)$ of vertices in $V_1 \times \ldots \times V_n$ such that for all protoconcept graph morphisms $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j$ $f(v_j) = v_i$ holds and $\rho_1^+(v_1) \times \ldots \rho_n^+(v_n) \cup \rho_1^-(v_1) \times \ldots \rho_n^-(v_n) \neq \emptyset$ holds.
   - The set of edges $E_{\mathfrak{C}}$ is the disjoint union $\dot{\bigcup}_{k=1}^{n} E_k$ factorized by the smallest equivalence relation containing the relation η_E which is defined by $(e_i, e_j) \in \eta_E$ iff there is a protoconcept graph morphisms $f : \mathfrak{G}_i \rightleftarrows \mathfrak{G}_j$ with $f_E(e_i) = e_j$.
   - The functions $\nu_{\mathfrak{C}}$, $\kappa_{\mathfrak{C}}$, and $\rho_{\mathfrak{C}}$ are defined exactly as in the case of distributed systems with semiconcept graphs.
2. The protoconcept graph morphisms $\gamma^i : \mathfrak{G}_i \rightleftarrows \mathfrak{C}$ are defined completely analogous to the semiconcept graph morphism in the case of distributed systems with semiconcept graphs. Because of the definition of $\mathcal{R}_{\mathfrak{C}}$ and the properties 4 and 5 $\gamma^i_{\mathcal{R}}$ maps $\doteq_i$ to $[\doteq_i]$ which is $\doteq_{\mathfrak{G}}$.

Theorem 2. *Let $\mathfrak{D}$ be a distributed system with protoconcept graphs satisfying the conditions 1-3. Then* $\lim(\mathfrak{D})$ *is the unique minimal cover of $\mathfrak{D}$ up to isomorphism and the core of* $\lim(\mathfrak{D})$ *is a protoconcept graph, i.e., a satisfiable quasi protoconcept graph.*

Again it is not possible to cover every distributed system which is not surprising because protoconcept graphs as well as semiconcept graphs include negation on the concept and on the relation level. Figure 6 pictures the core of the distributed system consisting of the protoconcept graphs in Figure 4 and the protoconcept graph morphisms in Figure 5. The common information of the system is that the chemical elements potassium and sodium which can be identified by their chemical symbols reacts with the compound water. Further ammonia and NaCl are no chemical element and do not react with helium and neon which are no compounds. Again for practical applications an isomorphic alphabet will be chosen.

6 Future Work

The theory of information flow by Barwise and Seligman now is extended to distributed systems including negation. For which applications protoconcept graphs are better than semiconcept graphs has yet to be explored. Here only syntactical constructions are given. The next step is the examination of the contextual semantics of these constructions. In particular this would be interesting for the core of the limit of a distributed system.

Further for concept graphs, semiconcept graphs, and protoconcept graphs over power context families the conceptual content can be considered (*cf.* [3], [5], and [18]). The conceptual content represents the information units of the concept graphs. One could imagine that the conceptual content of the core of the limit of a distributed is less or equal than the conceptual content of the concept graphs of the system if the core would be a concept graph over the same power context family. But unfortunately in the construction of the core a new alphabet is constructed, too. The hope is to find an embedding to the starting power context families in order to compare the conceptual contents.

For semiconcept graphs and protoconcept graphs extensions are defined which allow variables as references (*cf.* [6] and [7]). This theory can surely be extended to semiconcept graphs and protoconcept graphs with variables.

References

1. J. Barwise, J. Seligman: Information Flow: The Logic of Distributed Systems. Cambridge University Press, Cambridge 1997.
2. B. Ganter, R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer Verlag, Berlin Heidelberg New York 1999.
3. J. Hereth Correia, J. Klinger. Protoconcept Graphs: The Lattice of Conceptual Content. To appear in the conference proceedings of ICFCA 2004 in Sydney, Australia, Springer, Berlin Heidelberg New York 2004.
4. J. Klinger: Simple Semiconcept Graphs: a Boolean Logic Approach. In: H.S. Delugach, G. Stumme (Eds.): Conceptual Structures: Broadening the Base. Springer, Berlin Heidelberg New York 2001, 115-128.
5. J. Klinger: Semiconcept Graphs: Syntax and Semantics. Diplomarbeit, FB Mathematik, TU Darmstadt 2001.

6. J. Klinger: Semiconcept Graphs with Variables. In: U. Priss, D. Corbett, G. Angelowa (eds.): Conceptual Structures: Integration and Interfaces. Springer, Berlin Heidelberg New York 2002, 382-369.
7. J. Klinger: The Logic System of Protoconcept Graphs. FB4-Preprint, TU Darmstadt 2004, to appear.
8. J.Klinger, B. Vormbrock: Contextual Boolean Logic: How did it develop? In B. Ganter, A. de Moor (eds.): Using Conceptual Structures: Contributions to ICCS 2003, Shaker Verlag, Aachen, 143-156.
9. G. Malik: Information Transfer across Simple Concept Graphs. In: U. Priss, D. Corbett, G. Angelowa (eds.): Conceptual Structures: Integration and Interfaces. Springer, Berlin Heidelberg New York 2002, 48-61.
10. G. Malik: Distributed Systems with Simple Concept Graphs. In: A. de Moor, W. Lex, B. Ganter (eds.): Conceptual Structures for Knowledge Creation and Communication. Springer, Berlin Heidelberg New York 2003, 257-270.
11. S. Prediger: Simple Concept Graphs: a logic approach. In M.-L. Mugnier, M. Chein (eds.): Conceptual Structures: Theory, Tools and Applications. Springer, Berlin Heidelberg New York 1998, 225-239.
12. S. Prediger: Kontextuelle Urteilslogik mit Begriffsgraphen: Ein Beitrag zur Restrukturierung der mathematischen Logik. Dissertation, FB Mathematik, TU Darmstadt 1998. Verlag Shaker, Aachen 1998.
13. W. Schröter, K.-H. Lautenschläger, H. Bibrack, A. Schnabel: Chemie: Nachschlagebücher für Grundlagenfächer. Fachbuchverlag Leipzig, 1986.
14. J.F. Sowa: Conceptual Structures: information processing in mind and machine. Addison-Wesley, Reading 1984.
15. R. Wille: Conceptual Graphs and Formal Concept Analysis. In: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (eds.): Conceptual Structures: Fulfilling Peirce's Dream. Springer, Berlin Heidelberg New York 1997, 290-303.
16. R. Wille: Boolean Judgement Logic. In H.S. Delugach, G. Stumme (eds.): Conceptual Structures: Broadening the Base. Springer, Berlin Heidelberg New York 2001, 115-128.
17. R. Wille: Existential Graphs on Power Context Families. In: U. Priss, D. Corbett, G. Angelowa (eds.): Conceptual Structures: Integration and Interfaces. Springer, Berlin Heidelberg New York 2002, 382-396.
18. R. Wille: Conceptual Content as Information - Basics for Contextual Judgement Logic. In: A. de Moor, W. Lex, B. Ganter (eds.): Conceptual Structures for Knowledge Creation and Communication. Springer, Berlin Heidelberg New York 2003, 1-15.

Negation in Contextual Logic

Léonard Kwuida[1]*, Andreja Tepavčević[2], and Branimir Šešelja[2]

[1] Institut für Algebra, TU Dresden
D-01062 Dresden
[2] Institute of Mathematics, University of Novi Sad
Trg D. Obradovića 4, 21000 Novi Sad

Abstract. This contribution discusses a formalization of the "negation of a concept". The notion of "concept" has been successfully formalized in the early eighties and led to the theory of *Formal Concept Analysis.* Boole (1815-1864) developed a mathematical theory for human thought based on signs and classes. The formalization of the negation of concepts is needed in order to develop a mathematical theory of human thought based on "concept as a basic unit of thought". Two approaches will be discussed: negation as a partial or as a full operation on concepts.

1 Introduction

The aim of Formal Concept Analysis[1] is among others to support human thinking. Rudolf Wille introduced *Contextual Logic* with the aim to extend Formal Concept Analysis to a wider field. The main challenge is the formalization of a negation of a concept. He proposed different approaches. In the first case the "negation of a concept" is not necessarily a concept (protoconcepts and semiconcepts)[2]. For the second case the "negation of a concept" should be a concept (weak negation and weak opposition, concept algebras). In [Kw04] I discussed the conditions under which a weak negation is a Boolean complementation (a negation). I also proved that each concept algebra has a Boolean part. In this contribution we prove that the negation can be considered as a partial operation (explicitly defined by a weak negation) on concepts (Proposition 2 and Proposition 3). Contrary to the general situation where a concept lattice can be represented by different contexts, it happens that, to every Boolean algebra (considered as concept algebra) can be assigned only one context (up to isomorphism), the contranominal scale. However each context which is not of this form can be enlarged in such a way that the hierarchy of concepts and the Boolean part of the old context are preserved (Theorem 2). These results are presented in sections 4 and 5. To prepare these sections, we will give in sections 2 and 3 the historical and philosophical backgrounds.

* I am grateful to the PhD-program GrK334 and Gesellschaft von Freunden und Förderern, TU Dresden, for the financial support.

[1] The reader is referred to [GW99] for Formal Concept Analysis basics

[2] See [Wi00] and [KV03]

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 227–241, 2004.

2 From Logic to Formal Concept Analysis

2.1 From Logic to Lattice Theory

In the first half of the nineteenth century, George Boole's attempt to formalize logic[3] in [Bo54] led to the concept of Boolean algebras[4]. Boole gave himself the task "to *investigate the fundamental laws of those operations of the mind by which reasoning is performed; to give expression to them in the symbolical language of calculus, and upon this foundation to establish the science of Logic and construct its method; to make that method itself the basis of a general method for the application of the mathematical doctrine of Probabilities; and finally, to collect from the various elements of truth brought to view in the course of these inquiries some probable intimations concerning the nature and constitution of the human mind"*. The main operations he encoded are conjunction, disjunction, negation, universe and nothing, for which he derived some laws. He elaborated this logic as a theory of symbolic operations applied to classes of objects. Charles Sanders Peirce (1839-1914) and Ernst Schröder (1841-1902) introduced the notion of a lattice at the end of nineteenth century as they were investigating the axiomatics of Boolean algebras. Independently Richard Dedekind (1831-1916) got the same concept while working on ideals of algebraic numbers.

The general development of lattice theory really started in the mid-thirties with the work of Garrett Birkhoff (1911-1996). Other mathematicians like Valére Glivenko, Karl Menger, John von Neumann, Oystein Ore, etc., contributed to the formation of lattice theory as independent mathematical subdiscipline.

2.2 Restructuring Lattice Theory: Formal Concept Analysis

Lattice theory became a successful subject in mathematics and attracted many researchers. But ***why develop lattice theory***? In [Wi82] the author made this observation: *"lattice theory today reflects the general status of current mathematics: there is a rich production of theoretical concepts, results, and developments, many of which are reached by elaborate mental gymnastics; on the other hand, the connections of the theory to its surroundings are getting weaker: the result is that the theory and even many of its parts become more isolated"*. This isolation did not affect only lattice theory, but many other sciences. Wille was influenced by a book of Harmut von Hentig [He72], in which he discussed the status of the humanities and sciences. It was then urgent to "restructure" theoretical

[3] Kant (1723-1804) considered Logic as "... a science *a priori* of the necessary laws of thinking, not, however, in respect of particular objects but all objects in general: it is a science, therefore of the right use of the understanding and of reason as such, not subjectively, i.e. not according to empirical (psychological) principles of how the understanding thinks, but objectively, i.e. according to *a priori* principles of how it ought to think".

[4] In [Bu00] Burris discussed the Boole's algebra of Logic and showed that this was not a Boolean algebra, thought it led to Boolean algebras.

developments in order to integrate and rationalize origins, connections, interpretations and applications. Wille understood "restructuring lattice theory" as *"an attempt to unfold lattice theoretical concepts, results, and methods in a continuous relationship with their surroundings*", with the aim "to *promote a better communication between lattice theorists and potential users of lattice theory".*

Even the pioneer did not neglect this aspect. For example Birkhoff wrote: "*lattice theory has helped to simplify, unify and generalize mathematics, and it has suggested many interesting problems.*" In a survey paper on "lattices and their applications" [Bi38], he set up a more general viewpoint for lattice theory: *"lattice theory provides a proper vocabulary for discussing order, and especially systems which are in any sense hierarchies".*

The approach to lattice theory outlined in [Wi82] is based on an attempt to reinvigorate the general view of order. He went back to the origin of the lattice concept in the nineteenth-century attempts to formalize logic, where the study of hierarchies of concepts played a central role. A *concept* is determined by its *extent* and its *intent;* the extent consists of all objects belonging to the concept while the intent is the multitude of all properties valid for all those objects. The *hierarchy* of concepts is given by the relation of "subconcept" to "superconcept", i.e. the extent of the subconcept is contained in the extent of the superconcept while, reciprocally, the intent of the superconcept is contained in the intent of the subconcept.

3 Contextual Logic

3.1 Contextual Attribute Logic

Contextual Boolean Logic has been introduced with the aim to support knowledge representation and knowledge processing. An attempt to elaborate a Contextual Logic started with "Contextual Attribute Logic" in [GW99a]. The authors considered "signs" as attributes and outlined how this correspondence may lead to a development of a Contextual Attribute Logic in the spirit of Boole. Contextual Attribute Logic focuses, in a formal context (G, M, I), on the formal attributes and their extents, understood as the formalizations of the extensions of the attributes. It deals with logical combination of and relation between attributes. This is a "local logic".

The logical relationships between formal attributes are expressed via their extents. For example they said that "an attribute m implies an attribute n if $m' \subseteq n'$", and that "m and n are incompatible if $m' \cap n' = \emptyset$". In order to have more expressivity in Contextual Attribute Logic the authors introduced compound attributes of a formal context (G, M, I) by using the operational symbols $\neg$, $\bigwedge$ and $\bigvee$:

- For each attribute m they defined its negation, $\neg m$, to be a compound attribute, which has the extent $G \setminus m'$. Thus g is in the extent of $\neg m$ if and only if g is not in the extent of m.

- For each set $A \subseteq M$ of attributes, they defined the conjunction, $\bigwedge A$, and the disjunction, $\bigvee A$, to be the compound attributes that have the extents $\bigcap\{m' \mid m \in A\}$ and $\bigcup\{m' \mid m \in A\}$ respectively.
- Iteration of the above compositions leads to further compound attributes, the extents of which are determined in the obvious manner.

Observe that the complement of the extent m' is imposed to be an extent. The new extents are generated by the family $\{m' \mid m \in M\}$ and are closed under complementation, arbitrary union and intersection. Moreover to each extent generated as indicated above an attribute is assigned. This corresponds to Boole's logic where signs are attributes and classes are extents generated by the family $\{m' \mid m \in M\}$ with respect to complementation, union and intersection. The negation of an attribute is not necessary an attribute of the initial context. The new context is a dichotomic context in which all concepts are attribute concepts.

3.2 Contextual Concept Logic

At the second stage a Contextual Concept Logic should be developed by mathematizing the doctrines of concepts, judgments and conclusions on which the human thinking is based, suggested Wille. He divided the development of a Contextual Logic in three parts: a "Contextual Concept Logic", a "Contextual Judgment Logic", and a "Contextual Conclusion Logic". In [KV03] the authors compared various approaches to Contextual Judgment Logic. We are more concerned by the first step. Wille introduced concept algebras in [Wi00] with the aim *"to show how a Boolean Concept Logic may be elaborated as a mathematical theory based on Formal Concept Analysis".*

To extend the Boolean Attribute Logic to a Boolean Concept Logic, the main problem is the negation, since the conjunction and "disjunction" can be encoded by the meet and join operation of the concept lattice[5]. How can you define a negation of a concept? To define a negation of a sign, Boole first set up a universe of discourse, then took the complement of the class representing the given sign and assigned to the class he obtained a sign that he called the negation of the given sign. Here the universe is encoded by 1 and nothing by 0. Doing an analogy with formal concepts, the first problem is that the class of extents and the class of intents need not be closed under complementation. To have a negation as an operation on concepts, you can take as negation of a concept *(A, B)* of a formal context (G, M, I), the concept generated by the complement of its extent, namely $((G \setminus A)'', (G \setminus A)')$. Although the *principle of excluded middle*[6] is satisfied, the *principle of contradiction* does not always hold. This is called the weak negation of the concept (A, B). On the intent side we obtain a weak opposition satisfying the *principle of contradiction* but not always the *principle of excluded middle.* A concept lattice equipped with these two operations is called **concept algebra**. We give its formal definition.

[5] See [St94]

[6] See Section 4

Definition 1. *Let* $\mathbb{K} := (G, M, I)$ *be a formal context. For a formal concept (A, B) its* **weak negation** *is defined by* $(A, B)^{\triangle} := \left((G \setminus A)'', (G \setminus A)'\right)$ *and its* **weak opposition** *by* $(A, B)^{\triangledown} := \left((M \setminus B)', (M \setminus B)''\right)$. *The* **concept algebra** *of* $\mathbb{K}$, *denoted by* $\mathfrak{A}(\mathbb{K})$, *is the algebra* $\left(\mathfrak{B}(\mathbb{K}); \wedge, \vee, ^{\triangle}, ^{\triangledown}, 0, 1\right)$, *where* $\wedge$ *and* $\vee$ *denote the meet and the join operations of the concept lattice* $\underline{\mathfrak{B}}(\mathbb{K})$.

We cannot expect all laws of negation to be fulfilled by a weak negation or a weak opposition. In this contribution we demonstrate how the choice of appropriate subsets can reconcile the mathematic idea and the philosophic idea of a negation. These operations satisfy the equations in Definition 2 below; (see [Wi00]).

Definition 2. *A* **weakly dicomplemented lattice** *is a bounded lattice L equipped with two unary operations* $^{\triangle}$ *and* $^{\triangledown}$ *called* **weak complementation** *and* **dual weak complementation,** *and satisfying for all* $x, y \in L$ *the following equations:*

(1) $x^{\triangle\triangle} \leq x$, *(1')* $x^{\triangledown\triangledown} \geq x$,
(2) $x \leq y \implies x^{\triangle} \geq y^{\triangle}$, *(2')* $x \leq y \implies x^{\triangledown} \geq y^{\triangledown}$,
(3) $(x \wedge y) \vee (x \wedge y^{\triangle}) = x$, *(3')* $(x \vee y) \wedge (x \vee y^{\triangledown}) = x$.

We call $x^{\triangle}$ *the* **weak complement** *of* x *and* $x^{\triangledown}$ *the* **dual weak complement** *of* x. *The pair* $(x^{\triangle}, x^{\triangledown})$ *is called the* **weak dicomplement** *of* x *and the pair* $(^{\triangle}, ^{\triangledown})$ *a* **weak dicomplementation** *on L. The structure* $(L, \wedge, \vee, ^{\triangle}, 0, 1)$ *is called a* **weakly complemented lattice** *and* $(L, \wedge, \vee, ^{\triangledown}, 0, 1)$ *a* **dual weakly complemented lattice.**

We give without proof some properties; (see [Kw04, Proposition 2]).

Proposition 1. *For any weak dicomplementation* $(^{\triangle}, ^{\triangledown})$ *we have*

(4) $(x \wedge x^{\triangle})^{\triangle} = 1$, *(4')* $(x \vee x^{\triangledown})^{\triangledown} = 0$,
(5) $x^{\triangle} \leq y \iff y^{\triangle} \leq x$, *(5')* $x^{\triangledown} \geq y \iff y^{\triangledown} \geq x$,
(6) $(x \wedge y)^{\triangle} = x^{\triangle} \vee y^{\triangle}$, *(6')* $(x \vee y)^{\triangledown} = x^{\triangledown} \wedge y^{\triangledown}$,
(7) $(x \wedge y)^{\triangle\triangle} \leq x^{\triangle\triangle} \wedge y^{\triangle\triangle}$, *(7')* $(x \vee y)^{\triangledown\triangledown} \geq x^{\triangledown\triangledown} \vee y^{\triangledown\triangledown}$.

4 Negation

4.1 Philosophical Backgrounds

The problem of negation is almost as old as philosophy. It has been handled by many philosophers, with more or less contradicting point of views or confusing statements in one side, and with some attempts to formalize it on the other side. Aristoteles (384-322 BC) considered "negation" as the opposite of "affirmation". What does "affirmation" mean? Even if we consider "affirmation" as all what we know or can represent we still need the meaning of "opposite". According to Georg Friedrich Meier (1718-1777) the negation is the representation of the absence of something. Therefore a negation can only be represented in mind. This point of view is shared by Wilhelm Rosenkrantz (1821-1874) who stressed that a pure negation exists only in thinking, and only as opposite of an affirmation. Up to now we still need a clear definition of the terms "opposite" and "affirmation". John Locke (1632-1704) had doubts on the existence of negative representation; according to him the "not" only means lack of representation.

Meister Eckhart (1260-1328) stated that each creature has a negation in himself. Although he did not mention how the negation is obtained from a given creature, this implied that each creature should possess a negation. This idea was not welcomed by all philosophers. For example Wilhelm Jerusalem (1854-1923) thought that only a judgment can be rejected, and not a representation, like Brentano (1838-1917) wished.

On the way to formalize negation, we can note the idea of Georg Hagemann (1832-1903) for whom "each negation is ... originally an affirmation of being different". Could we consider each object A different of B as a negation of B? Not really. But at least each creature should be different from its negation. An even more "formal" definition is given by Adolph Stöhr (1855-1921). He said that *not-A* is a derived name from A according to the type of opposing derivation meaning what remains after removing A. Coming back to George Boole, he understood a negated sign as the representation of the complement of the class represented by the original sign in the given universe of discourse. The opposing derivation in this case is simply "taking the complement".

For a concept the negative is generally considered as the opposite or contrary of the positive and means the lacking of such properties. The trend is to assign to a negative concept an intent with negative attributes.

4.2 Some Properties of a Negation

Is there really a formal definition of "negation" ? The formalization of the negation by Boole and other operations of human thought led to Boolean algebras. The negation is encoded by a unary operation which satisfies some nice properties. For example it is an antitone involution, a complementation satisfying the de Morgan laws, and other properties. What are the properties that characterize a negation? In the philosophy some "necessary" conditions have been mentioned.

principium exclusi tertii : A or *not-A* is true. This is called *principle of excluded middle.* Thus an operation abstracting a negation should be a **dual-semicomplementation.** This principle is not accepted by all logicians. It is, for example, rejected by intuitionist logicians.

principium contradictionis : A and *not-A* is false. This is called *principle of contradiction.* Thus an operation abstracting a negation should be a **semicomplementation.**

duplex negatio affirmat : *not-not-A* has the same logical value as A. This is the *law of double negation.* A double negation is an affirmation. Thus an operation abstracting a negation should be an involution.

To these principles we can add the *de Morgan*[7] *laws* which help to get a negation of complex concepts. A brief brief history on the discussion on Logic can be

[7] De Morgan (1806-1871)

found in the paper “19th Century Logic between Philosophy and mathematics” by Volker Peckhaus [8].

4.3 Laws of Negation and Concept Algebras

In this part we investigate some subsets corresponding to the laws or principles mentioned in Subsection 4.2. L denotes a concept lattice $\underline{\mathfrak{B}}(G,M,I)$.

Concepts with negation. If there is a negation on a context $\mathbb{K}$ it should not depend on the intent-side or extent-side definition. Therefore the two unary operations should be equal.

$$(A,B)^{\triangle} = (A,B)^{\triangledown} \iff (A,B)^{\triangle} \leq (A,B)^{\triangledown} \iff G \setminus A \subseteq (M \setminus B)'$$

Thus any object g not in A has all attributes not in B. Each concept algebra in which the two unary operations coincide is said to be **with negation**. In [Kw04] it is shown that doubly founded concept algebras with negation are Boolean algebras. In addition the subset of elements with negation,

$$B(L) := \{x \in \mathfrak{B}(G,M,I) \mid x^{\triangle} = x^{\triangledown}\},$$

is a Boolean algebra. It is also a subalgebra of L. If the equality $^{\triangle} =^{\triangledown}$ does not hold on the whole concept algebra, we can consider the negation as a partial operation of L defined on *B(L)*. Its domain is quite often small although at least top and bottom element are in *B(L)*.

Law of Double Negation: Skeletons

Definition 3. *The set S(L) of elements that satisfy the law of double negation with respect to the weak opposition is called the* **skeleton** *and the set $\bar{S}(L)$ of elements that satisfy the law of double negation with respect to the weak negation is called the* **dual skeleton** *of L.*

$$S(L) := \{x \in L \mid x^{\triangledown\triangledown} = x\} \quad \text{and} \quad \bar{S}(L) := \{x \in L \mid x^{\triangle\triangle} = x\}.$$

We define the operations $\sqcap$ and $\sqcup$ on L by:

$$x \sqcap y := (x^{\triangledown} \vee y^{\triangledown})^{\triangledown} \quad \text{and} \quad x \sqcup y := (x^{\triangledown} \wedge y^{\triangledown})^{\triangledown}.$$

These operations are from $L \times L$ onto *S(L)*. Dually the operations

$$x \bar{\sqcap} y := (x^{\triangle} \vee y^{\triangle})^{\triangle} \quad \text{and} \quad x \underline{\sqcup} y := (x^{\triangle} \wedge y^{\triangle})^{\triangle}$$

are from $L \times L$ onto$\bar{S}(L)$. An ortholattice is a bounded lattice with an antitone complementation which is an involution.

[8] http://www.phil.uni-erlangen.de/~p1phil/personen/peckhaus/texte/logic_phil_math.html

Proposition 2. $(S(L),\wedge,\sqcup,{}^{\triangledown},0,1)$ *and* $(\bar{S}(L),\bar{\sqcap},\vee,{}^{\triangle},0,1)$ *are ortholattices.*

Proof. Trivially $(\bar{S}(L),\bar{\sqcap},\underline{\sqcup},0,1)$ is a bounded lattice. Moreover, for all x and y in $\bar{S}(L)$ we have $(x\bar{\sqcap}y)^{\triangle}=(x^{\triangle}\vee y^{\triangle})^{\triangle\triangle}=(x\wedge y)^{\triangle\triangle\triangle}=(x\wedge y)^{\triangle}$ and

$$x^{\triangle}\underline{\sqcup}y^{\triangle}=(x^{\triangle\triangle}\wedge y^{\triangle\triangle})^{\triangle}=(x\wedge y)^{\triangle}=(x\bar{\sqcap}y)^{\triangle}.$$

Similarly, $(x\underline{\sqcup}y)^{\triangle}=(x^{\triangle}\wedge y^{\triangle})^{\triangle\triangle}=(x^{\triangle\triangle}\vee y^{\triangle\triangle})^{\triangle}=(x\vee y)^{\triangle}$ and

$$x^{\triangle}\bar{\sqcap}y^{\triangle}=(x^{\triangle\triangle}\vee y^{\triangle\triangle})^{\triangle}=(x\vee y)^{\triangle}=(x\underline{\sqcup}y)^{\triangle}.$$

In addition $x\bar{\sqcap}x^{\triangle}=(x^{\triangle}\vee x^{\triangle\triangle})^{\triangle}=1^{\triangle}=0$ and $x\underline{\sqcup}x^{\triangle}=(x^{\triangle}\wedge x^{\triangle\triangle})^{\triangle}=1$. For x and y in $\bar{S}(L)$, we have $x\underline{\sqcup}y=(x^{\triangle}\wedge y^{\triangle})^{\triangle}=x^{\triangle\triangle}\vee y^{\triangle\triangle}=x\vee y$. The proof for $(S(L),\wedge,\sqcup,{}^{\triangledown},0,1)$ is obtained similarly. □

Remark 1. Skeleton and dual skeleton both contain all elements with negation. The meet operation and the join operation have been slightly modified on the dual skeleton and the skeleton respectively. The skeleton or dual skeleton can be neither distributive nor uniquely complemented even if the lattice L were distributive. Although the operation ${}^{\triangle}$ on the skeleton is a complementation, it is no longer a weak complementation. The condition

$$(x\wedge y)\vee(x\wedge y^{\triangle})=x$$

is violated. However the operation ${}^{\triangle}$ on the skeleton is an antitone involution that satisfies the de Morgan laws and is a complementation. Such operation is sometimes called "syntactic negation".

S(L) is generally not a sublattice of *L*. For a doubly founded lattice *L,* the skeleton is a sublattice of *L* if and only if it is a Boolean algebra. In particular *S(L)* $=L$ if and only if $(L,\wedge,\vee,{}^{\triangledown},0,1)$ is a Boolean algebra[9]·

Principle of contradiction/principle of excluded middle. The weak negation satisfies the principle of excluded middle. But the principle of contradiction fails. If we assume this principle for a weak negation we would get for all $x\in L$,

$$x=(x\wedge x^{\triangle})\vee(x\wedge x^{\triangle\triangle})=x\wedge x^{\triangle\triangle}$$

and $\bar{S}(L)=L$. In the case L is doubly founded we get a Boolean algebra. Instead of assuming this principle on the whole concept algebra, we can look for concepts on which the weak negation respects it. These concepts are automatically complemented. Their extent complement is again an extent. Hence the question arises if there is any characterization of concepts which extent complements are again extents. We denote by E_c the set of those concepts of a formal context $\mathbb{K}$.

$$E_c:=\{(A,B)\in\mathfrak{B}(\mathbb{K})\mid(G\setminus A,(G\setminus A)')\in\mathfrak{B}(\mathbb{K})\}$$

[9] A proof is provided in [Kw04]

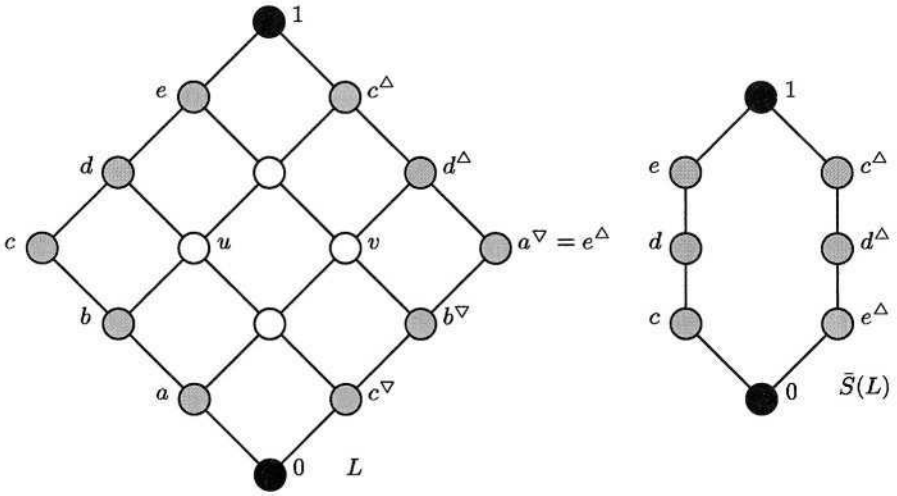


Fig. 1. A dicomplementation on the product of two 4 element chains and their skeletons. The corresponding context is $(J(L) \cup \{u,v\}, M(L) \cup \{u,v\}, \leq)$. The skeleton and dual skeleton are isomorphic. *B(L)* is the two element Boolean algebra. If we reduce this context its skeletons will be isomorphic to *B(L)* which is, in this case, a 4 element Boolean algebra.

Proposition 3. *The following assertions are valid:*

(i) $E_c = \{x \in \mathfrak{B}(\mathbb{K}) \mid x \wedge x^{\triangle} = 0\}$.
(ii) $E_c \subseteq \bar{S}(\mathfrak{A}(\mathbb{K}))$.
(iii) *If* $x \in E_c$ *then* $x^{\triangle}$ *is the pseudocomplement*[10] *of* x.
(iv) *Moreover if* $\underline{\mathfrak{B}}(\mathbb{K})$ *is distributive then*
  (a) E_c *is a sublattice of* $\underline{\mathfrak{B}}(\mathbb{K})$,
  (b) E_c *is a sublattice of* $\bar{S}(\mathfrak{A}(\mathbb{K}))$ *and*
  (c) E_c *is a Boolean algebra.*

Proof. (i) Let $x \in E_c$. We have $x^{\triangle} = ((G \setminus A)'', (G \setminus A)') = (G \setminus A, (G \setminus A)')$. Thus $x \wedge x^{\triangle} = 0$. Conversely if $x \wedge x^{\triangle} = 0$ for an $x \in \mathfrak{B}(\mathbb{K})$ then $A \cap (G \setminus A)'' = \emptyset$ and $(G \setminus A)'' \subseteq G \setminus A$. This means that $(G \setminus A)'' = G \setminus A$ and $x \in E_c$.

(ii) $x \in E_c \implies x = (x \wedge x^{\triangle}) \vee (x \wedge x^{\triangle\triangle}) = x \wedge x^{\triangle\triangle}$. Thus $x \in E_c \implies x = x^{\triangle\triangle}$.

(iii) $x \in E_c \implies x \wedge x^{\triangle} = 0$. If $x \wedge y = 0$ then $y = (x \wedge y) \vee (x^{\triangle} \wedge y) = x^{\triangle} \wedge y$ and $y \leq x^{\triangle}$. Thus $x^{\triangle}$ is the pseudocomplement of x.

(iv) We assume that $\underline{\mathfrak{B}}(\mathbb{K})$ is distributive.

[10] The pseudocomplement of x in $\underline{\mathfrak{B}}(\mathbb{K})$ (if it exists) is the largest element of the set $\{y \in \mathfrak{B}(\mathbb{K}) \mid y \wedge x = 0\}$

(a) Let $x, y \in E_c$. The assertion $(x \vee y) \wedge (x \vee y)^{\triangle} \leq (x \vee y) \wedge x^{\triangle} \wedge y^{\triangle} = 0$ is valid and implies that $x \vee y \in E_c$. It also holds:

$$(x \wedge y) \wedge (x \wedge y)^{\triangle} = (x \wedge y) \wedge (x^{\triangle} \vee y^{\triangle}) = 0.$$

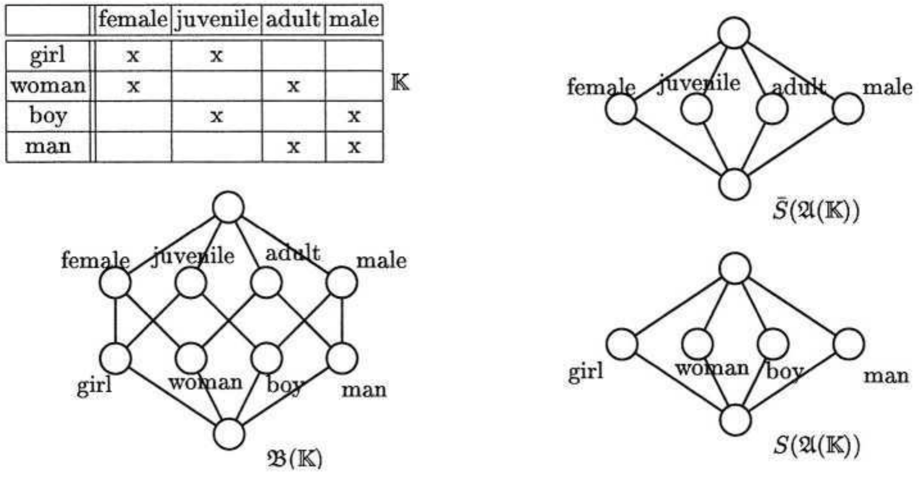
Thus E_c is a sublattice of L.

(b) To prove that E_c is a sublattice of $\bar{S}(L)$ it remains to show that $\bar{\sqcap}$ is the restriction of $\wedge$. This is immediate since for any x and y in E_c we have

$$x \bar{\sqcap} y = (x^{\triangle} \vee y^{\triangle})^{\triangle} = (x \wedge y)^{\triangle\triangle} = x \wedge y$$

(c) E_c is a complemented sublattice of a distributive lattice and thus a Boolean algebra.

□



	female	juvenile	adult	male
girl	x	x		
woman	x		x	
boy		x		x
man			x	x

𝕂
female
juvenile
adult
male
girl
woman
boy
man
𝔅(𝕂)
$\bar{S}(\mathfrak{A}(\mathbb{K}))$
$S(\mathfrak{A}(\mathbb{K}))$

Fig. 2. Complemented extents.

If we assume that one of the three above mentioned principles (see Subsection 4.2) holds in a concept algebra, we automatically get the others since the unary operation is forced to be a Boolean algebra complementation. If we consider only the elements satisfying the law of double negation (i.e. the skeleton), we get an ortholattice. Again here all the above mentioned laws hold. Thus if we want to work on the same context, we can consider a negation as a partial operation defined only for concepts of the skeleton.

De Morgan laws. The weak negation $^{\triangle}$ satisfies the meet de Morgan law. But the join de Morgan law fails.

Proposition 4. *if we assume the join de Morgan law for the weak negation then the dual skeleton is a complemented sublattice ofL.*

Proof. We assume the join de Morgan law for the weak negation. That is

$$(x \vee y)^{\triangle} = x^{\triangle} \wedge y^{\triangle}.$$

From $x \vee x^{\triangle} = 1$ we get $x^{\triangle} \wedge x^{\triangle\triangle} = 0$. Since all elements of the dual skeletons are of the form $x^{\triangle}$ and $x^{\triangle} \vee x^{\triangle\triangle} = 1$ it follows that the dual skeleton is complemented. We are going to prove that $\bar{S}(L)$ is a sublattice of L. Let x and y be elements of $\bar{S}(L)$. By the join de Morgan law we get

$$x \bar{\bar{\sqcap}} y = (x^{\triangle} \vee y^{\triangle})^{\triangle} = x^{\triangle\triangle} \wedge y^{\triangle\triangle} = x \wedge y.$$

Thus $x \wedge y$ belongs to $\bar{S}(L)$. We have already seen that $x \vee y$ belongs to $\bar{S}(L)$. Thus $\bar{S}(L)$ is a sublattice of L. □

Dually the weak opposition $^{\triangledown}$ satisfies the join de Morgan law. But the meet de Morgan law fails. If we assume the meet de Morgan law for the weak opposition then we will get the skeleton as a complemented sublattice of L.

5 Embeddings into Boolean Algebras

In this section we discuss how a context can be enlarged to get a negation as full operation on concepts. An element $a \in L$ is said to be $^{\triangle}$**-compatible** if for all $x \in L$, we have $a \leq x$ or $a \leq x^{\triangle}$. Dually is defined the notion of $^{\triangledown}$**-compatible element**.

5.1 Distributive Concept Algebras

For every set S the context $(S, S, \neq)$ is reduced. The concepts of this context are precisely the pairs $(A, S \setminus A)$ for $A \subseteq S$. Its concept lattice is isomorphic to the power set lattice of S. How does its concept algebra look like? For each concept $(A, S \setminus A) \in \mathfrak{B}(S, S, \neq)$ we have

$$(A, S \setminus A)^{\triangle} = ((S \setminus A)'', (S \setminus A)') = (S \setminus A, A)$$

and

$$(A, S \setminus A)^{\triangledown} = (A', A'') = (S \setminus A, A) = (A, S \setminus A)^{\triangle}.$$

The operations $^{\triangle}$ and $^{\triangledown}$ are equal. Thus $\left(\mathfrak{B}(S, S, \neq), \wedge, \vee, ^{\triangle}, \emptyset, S\right)$ is a Boolean algebra isomorphic to the powerset algebra of S. To get the converse we make use of the following fact:

Lemma 1.

(i) *If a and b are incomparable elements of a weakly dicomplemented lattice L such that none of them has* 1 *as weak complement then $a \vee b$ cannotbe $^{\triangle}$-compatible.*

(ii) *If $a \leq c$ and a is not $^{\triangle}-$compatible then c is not $^{\triangle}-$compatible.*

(i)' *Dually if a and b are incomparable elements of a weakly dicomplemented lattice L such that none of them has 0 as dual weak complement then $a \wedge b$ cannotbe $^{\triangledown}-$compatible.*

(ii)' *If $a \geq c$ and is not $^{\triangledown}-$compatible then c is not $^{\triangledown}-$compatible.*

Proof. We set $c := a \vee b$. Obviously $c \nleq a$. If $c \leq a^{\triangle}$ this would imply $a \leq c \leq a^{\triangle}$ and $a^{\triangle} = 1$ which is a contradiction, and (i) is proved. For (ii), note that if $a \nleq x$ and $a \nleq x^{\triangle}$ for some x then obviously $c \nleq x$ and $c \nleq x^{\triangle}$. This proves (ii). The rest is obtained analogously. □

We obtain

Theorem 1. *If $\mathbb{K}$ is a clarified context with no empty line or full column and such that $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangle}, 0, 1)$ and $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangledown}, 0, 1)$ are Boolean algebras then there is a set S such that $\mathbb{K}$ is isomorphic to $(S, S, \neq)$. In this situation the Boolean algebras $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangle}, 0, 1)$ and $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangledown}, 0, 1)$ are isomorphic.*

Proof. The standard context of the lattice $L := \underline{\mathfrak{B}}(\mathbb{K})$ is $(J(L), M(L), \leq)$ with *J(L)* the set of atoms and *M(L)* the set of coatoms. Note that $|M(L)| = |J(L)|$. Moreover the mapping $i : a \mapsto a'$ is a bijection from *J(L)* onto *M(L)* such that

$$a \leq b \iff b \neq i(a) \quad \forall a \in J(L) \text{ and } b \in M(L)$$

Therefore the context $(J(L), M(L), \leq)$ is isomorphic to $(J(L), J(L), \neq)$ by identifying each element $i(a) \in M(L)$ with $a \in J(L)$. We denote by S the set of irreducible objects of the context $\mathbb{K} := (G, M, I)$. If $G \neq S$ then there is an object $g \in G$ such that $\gamma g \geq \gamma g_1 \vee \gamma g_2$ with g_1 and g_2 in S. Note that

$$x \in S \implies (\gamma x)^{\triangle} \neq 1.$$

By Lemma 1 the element g is not $^{\triangle}-$compatible. Thus $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangle}, 0, 1)$ cannot be isomorphic to $(\mathfrak{B}(S, S, \neq), \wedge, \vee, ^{\triangle}, 0, 1)$. This contradicts the assumption that $(\mathfrak{B}(\mathbb{K}), \wedge, \vee, ^{\triangle}, 0, 1)$ is a Boolean algebra. The similar argument using the dual of Lemma 1 proves that $\mathbb{K}$ is also attribute reduced. □

Corollary 1. *Clarified contexts without empty line and full column with negation are exactly those isomorphic to $(S, S, \neq)$ for some set S.*

In general contexts are not reduced. To define a negation on a context $\mathbb{K}$, we can first reduce $\mathbb{K}$. If its reduced context is a copy of $(S, S, \neq)$ for a certain set S, then we are done. We define a negation on $\mathbb{K}$ by taking the concept algebra of $(S, S, \neq)$. In this case the set of concepts with negation is the whole concept lattice, which is a Boolean lattice. If $\underline{\mathfrak{B}}(\mathbb{K})$ is not a Boolean lattice, we might assume that our knowledge is not enough to get a negation. One option might be to extend the context to a larger one in which all concepts will have a negation. We should however make sure that doing so does not alter the relationship between concepts having a negation. This means that concepts having a negation in the old context should have the same negated concept in the extended context.

Let us examine the distributive case. To each distributive lattice can be assigned a context $(P, P, \not\geq)$ where $(P, \leq)$ is a poset. In the context $(P, P, \neq)$ all concepts have a negation. This context is obtained by extending the relation $\not\geq$ to $\neq$ on P. Of course $\not\geq$ is a closed subrelation of $\neq$. Therefore $\underline{\mathfrak{B}}(P, P, \not\geq)$ is a sublattice of $\underline{\mathfrak{B}}(P, P, \neq)$. It remains to verify that the negation is preserved on concepts with negation. This is straightforward since each concept with negation has a complement. In the extended context the complementation is unique and is at the same time the negation. Thus we are again able to define a negation on each context whose concept lattice is distributive. Unfortunately we cannot expect to have a lattice embedding from a nondistributive lattice into a Boolean algebra. An alternative is to find an embedding preserving the hierarchy of concept and the negation on concepts with negation. This can be done using a set representation of lattices.

5.2 General Case: Order Embedding

Unless otherwise stated, the lattices considered here are assumed to be doubly founded. Let L be a weakly dicomplemented lattice. Since a concept is determined by its intent and its extent, the negation of a context, if there is one, should not depend on the intent- or extent-side definition. i.e. the two weak operations should coincide. Recall that a concept is said with negation if its weak negation and its weak opposition coincide. The set *B(L)* of elements with negation forms a Boolean algebra which is a sublattice of *L.* The canonical context of (the weakly dicomplemented lattice) *B(L)* is (isomorphic to) a subcontext representing *B(L).*

Theorem 2. *Each doubly founded weakly dicomplemented lattice L can be order embedded in a Boolean algebra* $(B, \wedge, \vee, ', 0, 1)$ *in such a way that the structure of elements with negation are preserved.*

Proof. The set *J(L)* of supremum irreducible elements of L is a supremum dense subset of *L.* Its powerset $\mathcal{P}(J(L))$ is a Boolean algebra. We are going to embed L into $\mathcal{P}(J(L))$. Wedefine i by $i(x) = {\downarrow}x \cap J(L)$. Trivially

$$i(x \wedge y) = i(x) \cap i(y).$$

Therefore i is order preserving. Moreover i is an injective mapping. In fact if $x \not\leq y$ then there is an $a \in J(L)$ such that $a \leq x$ and $a \not\leq y$. Thus $a \in i(x)$ and $a \notin i(y)$. Therefore $i(x) = i(y) \implies x = y$. To prove that i is an order embedding we have to show that $x \leq y \iff i(x) \leq i(y)$. We assume that $i(x) \leq i(y)$. We get $i(x) = i(x) \cap i(y) = i(x \wedge y)$. Since i is injective we get $x = x \wedge y$ and $x \leq y$. Thus i is an order embedding.

It remains to prove that the weakly dicomplemented lattice operations on *B(L)* are preserved. Note that $i(0) = \emptyset$, $i(1) = J(L)$ and $i(x \wedge y) = i(x) \cap i(y)$. Let $x \in B(L)$. We have $\emptyset = i(0) = i(x \wedge x^{\triangledown}) = i(x \wedge x^{\vartriangle}) = i(x) \cap i(x^{\vartriangle})$. This equality implies that $i(x^{\triangle}) \subseteq i(x)'$. To prove the converse inclusion we consider an element a in $i(x)'$. Therefore $a \in J(L) \setminus i(x) = J(L) \setminus \downarrow x$. Then $a \leq x^{\triangle}$. i.e $a \in J(L) \cap \downarrow x^{\triangle} = i(x^{\triangle})$. Thus $i(x^{\nabla}) = i(x^{\triangle}) = i(x)'$ and the weak operations restricted on *B(L)* are preserved. For the join we have

$$i(x \vee y) = i((x \vee y)')' = i(x' \wedge y')' = (i(x') \cap i(y'))' = (i(x)' \cap i(y)')' = i(x) \cup i(y).$$

Therefore $i(B(L))$ is a Boolean algebra isomorphic to *B(L)*. In other words the structure of elements with negation is preserved by the order embedding. □

Remark 2. The similar construction holds with the set *M(L)* of meet irreducible elements and the mapping $j\colon x \mapsto j(x) := \uparrow x \cap M(L)$ from L into $\mathcal{P}(M(L))$. If L is a distributive lattice then *M*(*L*) is isomorphic to *J(L)* and $\mathcal{P}(M(L))$ is isomorphic to $\mathcal{P}(J(L))$. In this case there is an isomorphism $\psi : \mathcal{P}(M(L)) \to \mathcal{P}(J(L))$ such that $\psi \circ j = i$.

If we do not assume the distributivity, it might happen that *M(L)* and *J*(*L*) are of different cardinality. Without lost of generality we can assume that $|M(L)| \leq |J(L)|$ holds. Therefore there exists an embedding $\phi : \mathcal{P}(M(L)) \to \mathcal{P}(J(L))$ such that $\psi \circ j = i$.

Remark 3. If $(^{\triangle_1},^{\nabla_1})$ and $(^{\triangle_2},^{\nabla_2})$ are weak dicomplementations on a bounded lattice L such that $(^{\triangle_1},^{\nabla_1})$ is finer than $(^{\triangle_2},^{\nabla_2})$ then for all $x \in L$ we have

$$x^{\nabla_2} \leq x^{\nabla_1} \leq x^{\triangle_1} \leq x^{\triangle_2}.$$

Thus $B_2(L) \subseteq B_1(L)$ and $D_2(L) \subseteq D_1(L)$. The finer a weak dicomplementation is, the larger its set of elements with negation is. In the case of doubly founded lattice the finest dicomplementation is induced by the context $(J(L), M(L), \leq)$. Unfortunately $S_1(L)$ and $S_2(L)$ can be incomparable. On Figure 1 the context is not reduced but gives the largest skeleton and dual skeleton. If we reduce that context the skeleton will be the four element Boolean algebra. If the context is $(L, L, \leq)$ (largest clarified context), then the skeletons will be a (copy of the) two element Boolean algebra. It would be nice to have a description of the context giving the largest skeleton.

6 Conclusion

In this contribution we have examined the effect of some *laws of negation* on the *weak negation* introduced by Rudolf Wille. It turns out that only few contexts (contranominal scales) are convenient to explicitly define a negation. On all other contexts an explicit negation should be considered as a partial operation. Its domain depends on the laws to whom we give the priority. It can be the Boolean part, the skeleton or dual skeleton, or the set of concepts with complemented extents/intents. It is rather seldom to have an explicit negation satisfying the principles of Subsection 4.2. An alternative might be to enlarge the context. Note that any orthocomplementation satisfies these laws. But it cannot be explicitly defined unless the corresponding ortholattice is a Boolean algebra. The properties of Subsection 4.2 are often considered as the laws of negation. Unfortunately they do not characterize the negation. Hence we still ask ourself if there is a definition or a characterization of the negation. For a concept the negative is generally considered as the opposite or contrary of the positive and means the lacking of such properties. The trend is to assign to a negative concept an intent with negative attributes. So far, there is no definite solution for the problem of negation.

References

[Bi38] G.Birkhoff. *Lattices and their applications.* Bull. Amer. Math. Soc. **44** 793-800 (1938).

[Bi70] G.Birkhoff. *What can lattices do for you?* in *Trends in lattice Theory.* J.C Abbott (Ed.) Van Nostrand-Reinhold, New York 1-40 (1970).

[Bo54] G. Boole. *An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities.* Macmillan 1854. Reprinted by Dover Publ., New york (1958).

[Bu00] S.Burris. *The laws of Boole's thought.* Preprint (2000). http://www.thoralf.uwaterloo.ca/htdocs/MYWORKS/PREPRINTS/aboole.pdf

[Ei29] R. Eisler. *Wörtebuch der Philosophischen Begriffe.* Mittler, Berlin (1929).

[GK04] B. Ganter & L. Kwuida. *Representable weak dicomplementations on finite lattices. Contributions to General Algebra* **14**, J. Heyn Klagenfurt 63-72 (2004).

[Gr71] G. Grätzer *Lattice theory. First concepts and distributive lattices. A* Series *of Books in Mathematics.* W. H. Freeman and Company. (1971).

[GW99] B. Ganter & R. Wille. *Formal Concept Analysis. Mathematical Foundations.* Springer (1999).

[GW99a] B. Ganter & R. Wille. *Contextual attribute logic* in W. Tepfenhart, W. Cyre (Eds) *Conceptual Structures: Standards and Practices.* LNAI **1640** Springer, Heidelberg 337-338 (1999).

[He72] H. von Hentig. *Magier oder Magister? Über die Einheit der Wissenschaft im Verständigungsprozezss.* 1. Aufl. Suhrkamp Frankfurt (1974).

[KV03] J.Klinger & B.Vormbrock. *Contextual Boolean logic: How did it develop?* in Bernhard Ganter & Aldo de Moor (Eds.) *Using Conceptual Structures.* Contributions to ICCS 2003. Shaker Verlag 143-156 (2003).

[Kw03] L. Kwuida. *Weakly dicomplemented lattices, Boolean algebras and double p-algebras.* Technical Report MATH-AL-05-2003 (2003).

[Kw04] L. Kwuida. *When is a concept algebra Boolean?* in Peter W. Eklund (Ed.) *Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004. Proceedings.* Springer LNAI **2961** 142-155 (2004).

[St94] G.Stumme. *Boolesche Begriffe.* Diplomarbeit. TH Darmstadt (1994).

[Wi82] R. Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts* in I. Rival (Ed.) *Ordered Sets.* Reidel 445-470 (1982).

[Wi00] R. Wille. *Boolean Concept Logic* in B. Ganter & G.W.Mineau (Eds.) ICCS 2000 *Conceptual Structures: Logical, Linguistic, and Computational Issues.* Springer LNAI **1867** 317-331 (2000).

Using Brandom's Framework to Do Peirce's Normative Science: Pragmatism as the Game of Harmonizing Assertions?

Mary Keeler

Center for Advanced Research Technology in the Arts and Humanities
University of Washington, Seattle
mkeeler@u.washington.edu

Abstract. I introduce Robert Brandom's *Inferentialism,* which he calls a "rationalist expressivist" form of pragmatism, relating it to C.S. Peirce's unfinished project of Normative Science with its method of pragmatism as the logic of abduction, and suggest how Brandom's notion of a game (which I call Harmonizing Assertions) might serve as an effective methodological instrument in conceptual structures research, for its ultimate challenge of the human-computer tool interface.

1 Introduction

Although Robert Brandom's work on the "normative character of concept use" is not obviously linked to C.S. Peirce's unfinished project of Normative Science, I find enough evidence to suggest that Brandom's theoretical and methodological approach is compatible with Peirce's in a way that might be useful to conceptual structures research, especially in treating the issues of intuition, nominalism, and automation. Those issues are now even more relevant than when Peirce conceived his normative science to study the evolution of intelligent habit-change [see Note 1]. If we hope to continue augmenting intelligence, Brandom's framework offers a means of examining those issues in that evolution. I have argued that conceptual structures research should be theoretically based in that inquiry [see 1], and this paper extends my argument by suggesting a way of beginning to *do* normative science as a form of game. The game mode of operation exemplifies the method that normative science requires, Peirce's pragmatism, which he came to call "the logic of abduction." I start with a familiar example of the circumstances to be explored (the human-computer interface), then sketch the rationale and major features of Brandom's unconventional approach (relating it to Peirce's theory, here and there), briefly present Brandom's notion of a game, and relate it to Peirce's pragmatic method at the end.

I began to consider the game idea when John Sowa mentioned in a note on the PORT discussion list [see 3: 238-41]: "the phrase 'intuitively obvious to the most casual observer' was used as a joke to indicate a serious difficulty. ... It is essential to have tools that are easy to learn, easy to use, and easy for multiple collaborators to update dynamically. An important challenge for the PORT community is to design or develop new methods for collaborating that are superior to the current haphazard

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 242–260, 2004.

techniques for 'leaving documents around in virtual space.'" Heather Pfeiffer concurred: "We need to come up with ways that are intuitive so that tools will be easier to learn and easy to use."

Peirce identifies intuition as a *second* intellectual capability: "besides ordinary experience which is dependent on there being a certain physical connection between our organs and the thing experienced, there is a second avenue of truth dependent only on there being a certain intellectual connection between our previous knowledge and what we learn in that way" [*CP* 5.341, 1868]. Nominalism disregards the second capability, "for nominalism arises from taking that view of reality which regards whatever is in thought as caused by something in sense, and whatever is in sense as caused by something without the mind" [*CP* 8.25, 1871]: intellect is a form of automation.

In an early effort to apply Peirce's theory to computer interface design, I recommended for *information technology* what we should now more seriously consider for *knowledge technology.*

> Interface philosophy must have a new conceptual framework adequate to specify how technology development might proceed to meet human communication needs beyond the word processor's familiar context. Without this framework, interface philosophy (based on the current interaction framework) must simply *assume* that "intuitive interaction" or [what I describe as] "effective transparency" of the medium will make human-like communication possible at the interface, without being able to explain why or how this experience occurs for users. The need to explain how this "mediation experience" occurs has become more obvious as the human-computer interface evolves toward the goal of something even more like human-human communication. [4: 293]

Brandom's *inferentialist* framework (an ensemble of Hegel's rationalism, logical expressivism, and pragmatism about conceptual norms) has some origins in common with Peirce's normative science [see 5: 33], to which Brandom adds assertibility theory (the notion that propositional contents expressed by declarative sentences are like moves in a game) [see 5: 186]. He does not envision Peirce's scientific philosophy, but his criticisms of modern philosophy are similar to Peirce's. He focuses on the cultural context of inferences; although he acknowledges that these activities "arise within a framework of a natural world," he is concerned with phenomena that Hegel called *Geist*: "Cultural products and activities become explicit as such only by their use" [5: 33]. Peirce's normative science also focuses here, proposing to be a science to study our virtual world of ideals, with a logic of intellectual concepts.

Within Brandom's framework, we can do the detailed semantic work necessary to understand and observe the distinctions that Peirce's normative science requires. His inferentialist view of logic makes it possible to examine the difference between "the expressive role distinctive of *normative* vocabulary and the expressive role distinctive of *intentional* or explicitly *representational* vocabulary" [5: 23]. He argues that his functional theory of concepts makes "their role in reasoning, rather than supposed origin in experience, their primary feature." In his *rationalist* framework, concepts are norms that determine "what counts as a *reason* for particular beliefs, claims, and intentions, whose content is articulated by the application of those concepts" [5: 25].

Brandom describes his *rationalist, expressivist, pragmatic* version of inferentialism as a way of thinking about the origin of knowledge, the role played by language and logic, and also about meaning, mind, and knowledge which is free of currently prevailing empiricist commitments [see 5: 35]. He opposes many of the large theoretical, explanatory, and strategic movements of Anglo-American philosophy in the twentieth century: "empiricism, naturalism, representationalism, semantic atomism, formalism about logic, and instrumentalism about the norms of practical rationality" [5: 23]. In spite of his disagreements with modern analytic philosophy's world view, he identifies affinities there:

> I take my expository and argumentative structure and the criteria of adequacy for having made a claim with a clear content, argued for it, and responsibly followed out its consequences resolutely from the Anglo-American tradition ... For what I am trying to do is in a clear and specific inferential sense make explicit what is implicit in various philosophically important concepts. Among the examples treated ... are concepts such as *conceptual content, logic, ought, reliable, singular term,* what is expressed by 'of' or 'about' of intentional directedness, and objectivity." [5: 31-2]

He hopes that his theoretical approach can enlarge the scope of analytic philosophy to highlight "the possibilities opened up by engaging in social practices of giving and asking for reasons, [thereby bringing us] closer to an account of being human that does justice to the kinds of consciousness and self-consciousness distinctive of us as *cultural,* and not merely *natural,* creatures" [5: 35]. Peirce's method of pragmatism or reasoned conduct also contrasts with analytic philosophy's narrow focus on logic and linguistic activity [see 6: 43-45].

Brandom's assessment of modern philosophy and his inferentialist response generally resemble Peirce's rationale and purpose for pragmatism in responding to the nominalist legacy in the modern tradition.

> What is the purpose of Pragmatism? It is expected to bring to an end those prolonged disputes ... which no observations of facts could settle, and yet in which each side claims to prove that the other side is wrong. Pragmatism maintains that in those cases the disputants must be at cross-purposes. They either attach different meanings to words, or else one side or the other (or both) uses a word without any definite meaning. What is wanted, therefore, is a method for ascertaining the real meaning of any concept, doctrine, proposition, word, or other sign. The object of a sign is one thing; its meaning is another. Its object is the thing or occasion, however indefinite, to which it is to be applied. Its meaning is the idea which it attaches to that object, whether by way of mere supposition, or as a command, or as an assertion. (MS 323, in *CP* 5.6 [1906])

Somewhat like Peirce's criticism of modern philosophy and its logic, Brandom frequently claims that his approach reverses the traditional central arguments, and his inferentialism might be understood as filling in some details of Peirce's envisioned normative science, especially his view of the role of normative logic. He adopts the pragmatist's basic methodological view: "that the point of the theoretical association of *meanings* with linguistic expressions is to explain the *use* of those expressions. (Semantics must answer to pragmatics.)" [5: 185]. Peirce says: "pragmatism is the doctrine that every conception is a conception of *conceivable* practical effects" [*CP* 5.1

Brandom's only reference to Peirce in his major work, *Making It Explicit,* attributes the pragmatist's emphasis on "the normative character of cognitive undertakings" to what he says is their "unfortunately obscured" acknowledgment of "the central kantian legacy," to which he adds in parentheses: "Peirce is, as so often, an exception" [7: 289]. And a footnote in the introduction to his more recent work, *Articulating Reasons,* remarks: "Peirce is, on this issue as on so many others, a more complicated case" [5: 206]. I conclude that severely limited access to Peirce's writings in comprehensive and coherent form is the reason for his apparent slight notice of Peirce's work [see Keeler 2003]. Nevertheless, Brandom's objective seems to be compatible with Peirce's: "an expressive view of the role of logic and its relation to the practices constitutive of rationality ... [which] holds out the hope of recovering for the study of logic a direct significance for projects that have been at the core of philosophy since its Socratic inception" [5: 77]. Only extensive work to correlate their views might verify compatibility, because Peirce's theory has been so often (understandably) misrepresented, and because Brandom explicitly takes a Kant-Hegel-Frege-Wittgenstein-Sellars track from which to build his inferentialism. That work is well beyond the scope of one paper, here I briefly indicate enough evidence to compare their pragmatic approaches.

2 The Role of Logic

Brandom proposes that the contents of concepts are determined by inferential relations, those of "material inference" (such as "Pittsburgh is west of Baltimore") are implicit, and those of formal inference are explicit. His inferentialism is an effort to develop Frege's early insight of the role of logic as an attempt to codify inferences, rather than as a search for a special kind of truth. He argues that logic does not have to be considered as formalists do, giving us special epistemic access to a kind of truth by proving certain kinds of claims; instead, inferentialist pragmatism "starts with a practical distinction between good and bad inferences, understood as a distinction between appropriate and inappropriate doings, and goes on to understand talk about truth" [5: 12]. That view places logic in a normative context, consistent with Peirce's semeotic logic as the third normative science, theoretically specified by aesthetics and ethics [see Keeler 2003]. They agree on the task of a normative logic as Brandom describes it:

> Logic is not properly understood as the study of a distinctive kind of formal inference. It is rather the study of the inferential roles of vocabulary playing a distinctive expressive role: codifying in explicit form the inferences that are implicit in the use of ordinary, nonlogical vocabulary. Making explicit the inferential roles of the logical vocabulary then can take the form of presenting patterns of inference involving them that are formally valid in the sense that they are invariant ... But that task is subsidiary and instrumental only. The task of logic is in the first instance to help us say something about the conceptual contents expressed by the use of nonlogical vocabulary, not to prove something about the conceptual contents expressed by the use of logical vocabulary.

> On this picture, formal proprieties of inference essentially involving logical vocabulary derive from and must be explained in terms of material properties of inference essentially involving nonlogical vocabulary rather than the other way around. Logic is accordingly not a canon or standard of right reasoning. It can help us make explicit (and hence available for criticism and transformation) the inferential commitments that govern the use of all our vocabulary, and hence articulate the contents of all our concepts. [5: 30]

Peirce clearly distinguishes a broader sense of logic, which need not significantly challenge Brandom's, in establishing the role of normative science to "make explicit" the evolutionary nature of conceptual habits of thought understood in terms of his sign theory.

> The term "logic" is unscientifically by me employed in two distinct senses. In its narrower sense, it is the science of the necessary conditions of the attainment of truth. In its broader sense, it is the science of the necessary laws of thought, or, still better (thought always taking place by means of signs), it is general semeiotic, treating not merely of truth, but also of the general conditions of signs being signs (which Duns Scotus called grammatica speculativa), also of the laws of the evolution of thought, which since it coincides with the study of the necessary conditions of the transmission of meaning by signs from mind to mind, and from one state of mind to another, ought, for the sake of taking advantage of an old association of terms, be called rhetorica speculativa, but which I content myself with inaccurately calling objective logic, because that conveys the correct idea that it is like Hegel's logic. The present inquiry is a logical one in the broad sense. [*CP* 1.444 (1902); see Note 3].

In identifying the role of logic as a normative science, Peirce relates the two senses: "Logic may be defined as the science of the laws of the stable establishment of beliefs. Then, exact logic will be that doctrine of the conditions of establishment of stable belief which rests upon perfectly undoubted observations and upon mathematical, that is, upon diagrammatical, or, iconic, thought" [*CP* 3.429 (1896); see Note 3]. Peirce and Brandom appear to disagree on logic's theoretical generality, but agree on its pragmatic (instrumentalist) role. Brandom's work emphasizes the expressive role of formal logical vocabulary for "making explicit the inferences that are implicit in the conceptual contents of nonlogical concepts" [5: 60-1], which resembles Peirce's emphasis on the dialogic requirement of logic [see Note 2], our "inner and outer worlds" [*CP* 5.487], and his account of Existential Graphs as a logical instrument [*CP* 4.429-4.431]. And Brandom's defense of material inference appears to correlate with Peirce's semeotic account in terms of iconic and indexical sign action, as the pre-symbolic grounds of conceptual content in his Speculative Grammar.

Peirce would find Brandom justifiably critical of many pragmatists for the mistake of identifying propositional contents with only the *consequences* of endorsing a claim: "looking downstream to the claim's role as a premise in practical reasoning and ignoring its proper antecedents upstream" [5: 66]. As a conceptual pragmatist, Brandom claims:

> the sense of endorsement that determines the force of assertional speech acts involves, at a minimum, a kind of *commitment* the speaker's *entitlement* to which is always potentially at issue. The assertible contents expressed by declarative sentences whose utterance can have this sort of force must accordingly be inferentially articulated along both normative dimensions. Downstream, they must have inferential consequences, commitment to which is entailed by commitment to the original content. Upstream, they must have inferential antecedents, relations to contents that can serve as premises from which entitlement to the original content can be inherited. [5: 194-95]

Foregoing more detailed comparison, I need only emphasize some significant points. Both Peirce and Brandom identify logic in both a narrow (formal) context and in a broad (normative) context, and both view logic as having the purpose of clarifying concepts. Brandom's role for logic in making explicit any implicit expressions in linguistic discourse is compatible with Peirce's arguments for logic as a normative science and pragmatism as its methodeutic. In that expressive role, Brandom says logic is "a distinctive set of tools for saying something that cannot otherwise be made explicit," making it possible to express a claim in conditional form [5: 19]. Furthermore, he says: "Logic is the linguistic organ of semantic self-consciousness and self-control. The expressive resources provided by logical vocabulary make it possible to criticize, control, and improve our concepts" [5: 149], by "making explicit the implicit background against which alone anything can be made explicit" [5: 35]. He insists that "'Assertion', 'claim', 'judgment', and 'belief are systematically ambiguous expressions," and that his pragmatism "seeks to explain what is assert*ed* by appeal to features of assert*ings,* what is *claimed* in terms of claim*ings,* what is jud*ged* by judg*ings,* and what is expressed by express*ings* of it—in general, the content by the act, rather than the other way around." His conceptual pragmatism approaches the contents of conceptually *explicit* propositions or principles from the direction of what is *implicit* in practices of using expressions and acquiring and deploying beliefs" [5: 4; see 23-24]. Brandom stresses that his inferentialism is not bound by what we can identify as the empiricist tradition's approach to meaning, mind, knowledge and action.

3 Brandom's Rationalist Semantic Theory

Most significant (and probably surprising) for conceptual structures researchers, Brandom's work correlates with Peirce's normative science in his arguments against the empiricist tradition (which Peirce criticizes as nominalist [see for example *CP* 8.20-21]). A central goal of Brandom's work is to introduce a way of freeing philosophy from that tradition in its distinctive twentieth-century form (developed by thinkers such as Russell, Carnap, and Quine) which perpetuates a "classical insistence on the origin of knowledge in experience and emphasis on the crucial cognitive role played by language and logic." He does not deny that "consideration of perceptual practices must play a crucial role in our epistemology and semantics," but he does deny the empiricist platitude that "without perceptual experience we can have no knowledge of contingent matter of fact, and more deeply, that conceptual content is unintelligible apart from its relation to perceptual experience" [5: 23].

Brandom's primary argument is with the underlying semantic theory, which grounds theoretical and practical reasoning, and concept use, in episodes occurring in immediate experience: "sense experiences on the cognitive side, and felt motivations or preferences on the active side" [5: 24]. He disagrees with those who think of having these experiences as not requiring the exercise of specifically conceptual abilities, but instead as a preconceptual capacity perhaps shared with other mammals, providing raw materials upon which conceptual activities operate. "We ought to respect the distinction between genuine perceptual *beliefs*—which require the application of *concepts*—and the reliable responses of minerals, mines, and matador fodder. I claim that an essential element of that distinction is the potential role as both premise and conclusion in reasoning (both theoretical and practical) that beliefs play" [5: 109; compare his concerns with Peirce's arguments for *perceptual judgment, CP* 4.539-41(1906) and *CP* 7.629 (1903)].

Brandom argues that our sapience can be identified in terms of inference and in terms of truth, and that the contents of both are distinguished as intelligible by their *propositional* form. "What we can offer as a reason, what we can take or make true, has a propositional content, a content of the sort that we express by the use of declarative sentences and ascribe by the use of 'that' clauses. Propositional contents stand in inferential relations, and they have truth conditions." He contends that such states as belief, desire, and intention are contentful in the sense that we can appropriately raise *pragmatic* questions asking: "under *what* circumstances what is believed, desired, or intended would be *true*" [5: 158]. His primary concern is to present a view of the relation between what is *said* or *thought* and what it is said or thought *about.* The former is the propositional dimension of thought and talk, and the latter is its *representational* dimension.

According to empiricist semantic theory, Brandom explains, "the content of experience, awareness, and knowledge is to be understood in the first instance in representational terms: as a matter of what is (or purports to be) represented by some representing states or episodes" [5: 24-5]. And in contemporary versions, he finds this "representational content" usually classified in terms of "what objects, events, or states of affairs actually causally elicited the representation, or which ones would reliably elicit representations of that kind under various conditions." The dominant philosophy of language begins with perceptual experience, then "focuses on reference, denotation, and extension, following a pattern of extensional model-theoretic semantics of the language of first-order predicate logic" [5: 24-5]. In contrast, his view accounts for knowing (or believing, or saying) that such and such is the case in terms of knowing *how* (being able) to *do* something [see 5: 4]. But he distinguishes his pragmatic view from that of American pragmatists, such as James and Dewey, by his pursuit of a *relational* linguistic approach, which he says "provides a counterbalance to the Frege-Russell-Carnap-Tarski platonistic model-theoretic, representational approach to meaning" [5: 6, see 59].

In contrasting the empiricist with the rationalist explanations of the origins of conceptual phenomena, Brandom finds the most telling difference in how they explain the relation between awareness and concept use: "Empiricism attempts to understand the content of concepts in terms of the origin of empirical beliefs in experience that we just find ourselves with, and the origin of practical intentions in desires or preferences that in the most basic case we just find ourselves with. The *rationalist* order of explanation understands concepts as norms determining what counts as a *reason* for particular beliefs, claims, and intentions, whose content is articulated by the application of those concepts [for which such statuses can be reasons]" [5: 25-6]. For concepts to play their role according to the empiricist, there must be prior awareness that justifies or makes the application of a particular concept appropriate, which "must amount to something more than just the reliable differential responsiveness of merely irritable devices such as land mines and pressure plates …" Brandom argues that a *functional rationalist* theory explains the primary role of concepts in reasoning, instead of their supposed origin in experience [see 5: 25-6; compare to [*CP* 2.302] and [*CP* 5.442].

Brandom confronts us with what he calls the inferentialist lesson: even noninferential reports (those associated with observable properties such as color) must be inferentially articulated, in order to have conceptual content. Without that requirement, we cannot distinguish between noninferential reporters and automatic machinery such as thermostats and photocells, which also reliably respond differentially to stimuli. Following Wilfred Sellars, he proposes: "for a response to have conceptual content is just for it to play a role in the inferential *game of making claims and giving and asking for reasons.* To grasp or understand a concept is to have practical mastery over the inferences it is involved in—to know, in the practical sense of being able to distinguish (a kind of know-how), what follows from the applicability of a concept, and what it follows from" [5: 108].

A parrot uttering sentences cannot be making a potential move in a game of giving and asking for reasons, which is applying a concept as node in an inferential network of other concepts. Brandom maintains that "mastery of *any* concepts, entails mastery of *many* concepts: grasp of one concept consists in mastery of at least some of its inferential relations to other concepts." Furthermore, "to be able to apply one concept *non*inferentially, one must be able to use others *inferentially.* For unless applying it can serve at least as a premise from which to draw inferential consequences, it is not functioning as a concept at all." The radical mistake, in his view then, is to think that there could be an autonomous language game, consisting entirely of noninferential reports: "The requirement is that for *any* concepts to have reporting uses, some concepts must have *non*reporting uses [whose only use is inferential]" [5: 49]. The *representational* dimension of propositional contents reflects the *social* structure of their *inferential* articulations in the game of giving and asking for reasons [5: 183].

The rationalist semantic theory Brandom introduces does not take *representation* as its fundamental concept. That methodological commitment, he says, "has the advantage of bringing into relief certain features of conceptual representation that are hard to notice otherwise ... [by] offering an account of referential relations to objects in terms ultimately of inferential relations among claims" [5: 28]. "The question I address is why any state or utterance that has propositional content also should be understood as having representational content. (For this so much as to be a question, it must be possible to characterize propositional content in nonrepresentational terms" [5: 158]. Brandom's inferentialism is committed to a semantic *holism,* in contrast to the *atomism* typical of representational semantics, and he recognizes difficulties for his rationalist semantic theory in facing problems both concerning the *stability* of conceptual contents, as beliefs change along with their commitments to the propriety of various inferences, and concerning the possibility of *communication,* when individuals endorse different claims and inferences [see 5: 29]. His *model of discursive practice* renders these concerns less troublesome, by treating concepts as *norms* determining the *correctness* of various moves in the game.

4 Brandom's Model of Discursive Practice: Defining the Normative Fine Structure of Rationality

Brandom summarizes his inferentialist model of discursive practice in three themes:

1. Conceptual content is to be understood in terms of roles in reasoning rather than exclusively in terms of representation.
2. The capacity for such reasoning is not to be identified exclusively with mastery of a logical calculus.
3. Besides theoretical and practical reasoning using contents constituted by their role in material inferences, there is a kind of expressive rationality that consists in making implicit content-conferring inferential commitments explicit as the contents of assertible commitments. In this way, the material inferential practices [that] govern and make possible the game of giving and asking for reasons are brought into that game, and so into consciousness, as explicit topics of discussion [see 5: 61].

Brandom offers his model of discursive practice as one way of thinking about the discursive commitments that express claims, making it possible to define *communication* and *justification* in the normative terms of the interaction of inferentially articulated *authority* and *responsibility.* In asserting something we express our *commitment,* giving the content our authority, and licensing others to undertake a corresponding commitment to use as a premise in their reasoning. One essential aspect, then, is *communication:* "the interpersonal, intracontent inheritance of entitlement to commitments." Also, in making an assertion we undertake a responsibility to justify the claim if appropriately challenged, thereby redeeming our *entitlement* to the commitment acknowledged by the claiming; so another essential aspect of this model of discursive practice is *justification:* "the intrapersonal, intercontent inheritance of entitlement to commitments" [5: 165].

If a claim expressed by one sentence entails the claim expressed by another, we treat anyone committed to the first as thereby committed to the second. Brandom reminds us that although we typically think of inferences solely in terms of the relation between premise and conclusion (as a monological relation among propositional contents), in discursive practice, the giving and asking for reasons involves both intercontent and interpersonal relations. He stresses: "the representational aspect of the propositional contents that play the inferential roles of premise and conclusion should be understood in terms of the social or dialogical dimension of communicating reasons, of assessing the significance of reasons offered by others," rather than as arrived at noninferentially from sense perception [5: 166]. Peirce stresses a similar semeotic understanding of how his Existential Graphs should be applied dialogically [8; *CP* 4.429-431].

To apply Brandom's model in any context, he reminds us, we must select *propositionally* contentful expressions that can serve both as premise and conclusion in inference: what can be offered as, and itself stand in need of, *reasons.* Understanding or grasping such propositional content is a kind of know-how, or practical mastery of the game of giving and asking for reasons: being able to tell what is a reason for what, distinguish good reasons from bad. To play such a game is to keep *score* on what other players are committed and entitled to, as two dimensions of normative status. Understanding the content of a claim or a belief is being able to accord it proper significance, or "knowing how it would change the score in various contexts" [5: 165-66].

Along with these two sorts of normative status, we must keep track of their interaction: "Thus commitment to the content expressed by the sentence 'The swatch is red' rules out entitlement to the commitment that would be undertaken by asserting the sentence 'The swatch is green'." For each sentence there will be a set of sentences that are incompatible with it, and furthermore: "Inclusion relations among these sets then correspond to inferential relations among the sentences. That is, the content of the claim expressed by asserting The swatch is vermilion' entails the content of the claim expressed by asserting 'The swatch is red,' because everything incompatible with being red is incompatible with being vermilion" [5: 193-94]. Brandom summarizes, explaining in terms that correspond to Peirce's stages of inquiry [see *CP* 7.469-72, 1908].

The two sorts of normative status that must be at play in practices that incorporate a game of giving and asking for reasons, commitment and entitlement, induce *three* sorts of inferential relations in the assertible contents expressed by sentences "suitably caught up in those practices":

> *committive* (that is, commitment-preserving) inferences, a
> category that generalizes deductive inference;
> *permissive* (that is, entitlement-preserving) inferences, a
> category that generalizes inductive inference; and
> *incompatibility* entailments, a category that generalizes
> modal (counterfactual-supporting) inference, [see 5: 194]

These three sorts of inferential consequence relations are ranked: "all incompatibility entailments are commitment-preserving (though not vice versa), and all commitment-preserving inferences are entitlement preserving (though not vice versa)" [5: 195]. Then inheritance of commitment, inheritance of entitlement, and entailments according to the incompatibilities defined by the interactions of commitments and entitlements compose the three axes that "inferentially articulate" what Brandom calls *normative fine structure of rationality,* to describe the rational practices that include the production and consumption of reasons, such as in the Sellarsian game of giving and asking for reasons [see 5: 195]. I will briefly introduce Brandom's notion of this game and then suggest how it might relate to doing Peirce's normative science.

5 The Game of Harmonizing Assertions

Peirce treated his pragmatism hypothetically (even suggesting that his EG could be used to prove that hypothesis [*CP* 4.533fn]), and Brandom treats his pragmatic linguistic rationalism hypothetically, proposing to explore its consequences in the context of the game. Peirce's pragmatism understands beliefs in terms of habits as conduct: "Now to be deliberately and thoroughly prepared to shape one's conduct into conformity with a proposition is neither more nor less than the state of mind called Believing that proposition, however long the conscious classification of it under that head be postponed" [*CP* 6.467]. "A belief is a habit; but it is a habit of which we are conscious. The actual calling to mind of the substance of a belief, not as personal to ourselves, but as holding good, or true, is a judgment. An inference is a passage from one belief to another; but not every such passage is an inference. ... In inference one belief not only follows after another, but follows from it" [*CP* 4.53].

Brandom begins with similar pragmatic understandings but elaborates in his rationalist terms: beliefs are commitments, commitments become concepts when they become explicit (by a rational process exemplified in the game of asking for and giving reasons) [see 5: 29]: concepts are norms (or socially articulated inferences), which we can express discursively as claims [see 5: 165]. Claims permit us to formulate commitments inferentially as assertions, to expose those beliefs that would otherwise remain implicit and unexamined in material concepts. In these logical formulations, we can then display the relevant grounds and consequences and assert their inferential relations, so that implicit inferential commitments in the content are open to challenges and demands for justification, "to groom and improve our inferential commitments, and so our conceptual contents" [5: 71]. When we use a concept, in Brandom's terms, we make a commitment to an inference from its grounds to its consequences of application, which justifies our conduct: "Acting with reasons is being *entitled* to one's practical commitments" [5: 93].

My designation of the game as "Harmonizing Assertions" is partly suggested by Brandom's brief reference to Michael Dummett's idea that a theory of meaning should account for the 'harmony' between circumstances and consequences in the application of concepts as we *ought* to employ them, to maintain a conservative extension of the concepts used in a language [see 7: 124-32]. Brandom prefers to apply the term at the metalanguage level of semantic theory, to explicate the role of logical vocabulary in the clarification of concepts under change, as in the evolution of technical concepts. Here, Brandom warns that when we use logic to display the relevant grounds and consequences and to assert their inferential relations, we must realize that clarification will never achieve complete transparency of commitment and entitlement, which must serve as an *ideal* modeled on Socratic practice. Brandom realizes that his notion of *harmony* is more complicated to explain than either empiricist or rationalist epistemologies allow us to think: as among matters of fact (expression of commitment as belief) or as relation of ideas (expression of commitment as meaning). He insists that these two options should not be considered exhaustive, or leave us with the puzzle that harmony must either be compelled by the facts or dictated by freely chosen meanings [see 5: 72-76].

Brandom contrasts the simplistic "either/or" controversy in modern philosophy of rationalism vs. empiricism with his framework: conceptual contents are conferred on expressions when they are "caught up in a structure of inferentially articulated commitments and entitlements," which the game is conceived to embody.

> [A]sserting a sentence is implicitly undertaking a commitment to the correctness of the material inference from its circumstances to its consequences of application ... Understanding or grasping a propositional content is here presented not as the turning on of a Cartesian light, but as practical mastery of a certain kind of inferentially articulated doing: responding differentially according to the circumstances of proper application. This is not an all-or-nothing affair. ... Thinking clearly is on this inferentialist rendering a matter of knowing what one is committing oneself to by a certain claim, and, what would

> entitle one to that commitment. Writing clearly is providing enough clues for a reader to infer what one intends to be committed to by each claim, and what one takes would entitle one to that commitment. Failure to grasp either of these components is failure to grasp the inferential commitment that use of the concept involves, and so failure to grasp its conceptual content [5: 63-64].

In thinking critically, Brandom stresses, we must examine our idioms to be sure that we are prepared "to endorse and so defend the appropriateness of the material inferential transitions" implicit in the concepts we employ—"they must not be allowed to remain curled up inside loaded phrases such as 'enemy of the people' or 'law and order'" [5: 70].

He concludes that the idea of a theory of semantic or inferential harmony makes sense only if it investigates the "ongoing elucidative process, of the 'Socratic method' of discovering and repairing discordant concepts." We can give the concept of *harmony* content only in "the process of harmonizing commitments, from which it is abstracted." He compares that process to judges formulating principles of common law: "intended both to codify prior practice, as represented by precedent, expressing explicitly as a rule what was implicit therein, and to have regulative authority for subsequent practice" [5: 75-76]. Expressing material inferential commitments explicitly has an essential role in the Socratic practice of harmonizing our commitments, and a commitment becomes explicit when it is "thrown into the game of giving and asking for reasons as something whose justification, in terms of other commitments and entitlements, is liable to question." Brandom stresses: "Any theory of the sort of inferential harmony of commitments we are aiming at by engaging in this reflective rational process must derive its credentials from its expressive adequacy to the practice before it should be accorded any authority over it" [5: 76].

Engaging in the game is testing Brandom's pragmatic inferentialist hypothesis that we can "aim at harmonizing our commitments" by making them explicit in claims, remembering that claims (as assertions) are essentially performances that can both serve as and stand in need of reasons, and their propositional contents are essentially what can serve as both premises and conclusions of inferences [see 5: 189]. He argues for two defining features.

1. A set of practices is recognizable as a game of giving and asking for reasons concerning assertions only if it involves acknowledging at least two sorts of normative status: commitments and entitlements, with some general structures relating them.
2. The commitments and entitlements that specify proposition contents display a particular sort of objectivity: they are not about any attitudes of the linguistic practitioners who produce and consume them as reasons [see 5: 190].

Brandom recognizes the challenge of objectivity for his theory: "to start with a notion of propriety of assertion that is grounded in and intelligible in terms of the practice of speakers and audiences, and yet which is rich enough to find normative assessments that are objective in the sense of transcending the attitudes of practitioners" [5: 198]. Peirce's normative science has the same challenge (and ultimate purpose): how to arrive at authentic objectivity under the normative conditions of fallibility explained in his semeotic logic. We can understand the objectivity of our thought, Brandom explains, in the way the contents of our thought go beyond the attitudes of endorsement or entitlement which we have toward those contents, as an aspect of the normative fine structure of rationality revealed in the game. We play the game of Harmonizing Assertions (keeping score on commitments, entitlements and their inferential relations) to reveal that normative fine structure of rationality—making explicit what is implicit in conceptual content.

Sentences are the counters or markers that add toward a score. Producing or playing one is making an assertional move in the game, making a claim, which requires knowing at least some of its consequences (what other moves one is committing to). Each counter bears a player's mark, is on that player's list, or is kept in the player's box, and each player's collection of these counters constitutes his score. Each player, at all times, must have a way of distinguishing those sentences he is prepared to assert, perhaps when prompted, from those he is not. And Brandom stresses that for a move to be recognized as an assertion, it must not be idle, "it must make a difference, it must have consequences for what else it is appropriate to do, according to the rules of the game" [5: 190-1]. Because assertions express beliefs, when a player plays a counter, or otherwise adds it to her score, that play commits her to playing others which add to her score.

Putting a sentence on your list of judgments, or putting it in your belief box, has consequences for how you ought, rationally, to act, judge, and believe. We may be able to construct cases where it is intelligible to attribute beliefs that are consequentially inert and isolated from any others, such as: "I just believe that cows look goofy, that's all." But nothing follows from that, no one is obliged to act in any particular way on that belief, and we could not intelligibly understand all our beliefs this way. If putting sentences onto a list (or into a box) never has consequences for what else belongs there, then the list ought not to be understood as consisting of all of someone's judgments or beliefs [see 5: 191]. Understanding the significance of an assertional move (which is a claim), requires that players understand at least some of its consequences, that they know what else (what other moves) they would be committing themselves to by making that claim. Brandom stresses the point that these are *rational* relations. (For further game specifications, see 7: chapters 3 and 4).

6 Conclusions

Brandom says his pragmatism "offers an account of knowing (or believing, or saying) *that* such and such is the case in terms of knowing *how* (being able) to *do* something" [5: 4]. His framework might enable Conceptual Structures research to *express* itself in a form through which it could apply its own technology, articulating its medium (conceptual structures) as its message (what that medium can do in tools). Brandom's ideal of harmony may not satisfy Peirce's theoretical aim, but his strategy heads toward normative science by insisting that logic's role is not to "trade primitive goodness of inference for the truth conditionals of the formalist approach to logic." We can use conditional inferences to make implicit material commitments explicit, but we must not assume that logical vocabulary by itself can make inferences *good* inferences [see 5: 86-87]. He identifies the pragmatic role of logic in normative terms:

> Logic transforms semantic practices into principles. By providing the expressive tools permitting us to endorse in what we say what before we could endorse only in what we did, logic makes it possible for the development of concepts by which we conceive our world and our plans (and so ourselves) to rise in part above the indistinct realm of mere tradition, of evolution according to the results of the thoughtless jostling of the habitual and the fortuitous, and enter the comparatively well lit discursive marketplace, where reasons are sought and proffered, and every endorsement is liable to being put on the scales and found wanting. [5: 153]

What if we could *do* the normative process of tool interface design by developing Brandom's notion of the game for that purpose? More significantly, what if interfaces could be developed along with application systems (involving needs assessment, requirements formulating, prototyping, testing, evaluation, and standardizing) within a "harmonizing" framework, rather than as empirically observed and described interactions, after systems are constructed? Furthermore, if the ultimate framework for comprehending human-computer communication must be in terms anything like human-human communication, then surely Conceptual Structures research to augment that development should be using (at least exploring) such a *dialogic* game framework? We unquestioningly accept empiricist assumptions in the prevailing model of an "interaction process." Perhaps we should consider Brandom's advice: "The more promising alternative is to focus to begin with on the conceptual articulation of perceptually acquired and practically pursued commitments and entitlements rather than the experiences and inclinations with which we simply find ourselves" (which we refer to as "intuition" and treat as no different than automatic responses, perpetuating naive nominalism). Brandom claims his strategy leads us to expect that we will "learn more about a building by studying blueprints than by studying bricks" [5: 32].

In the short history of interface research (which we might date from Douglas Engelbart's work in the 1960s [http://sloan.stanford.edu/MouseSite/]), "user testing" has been conducted according to empirical science principles and assumptions. Some of us who have examined that research, can agree with Brandom that the dominant modern tradition of empiricism has systematically slighted "the distinctive nature, contribution, and significance of the *conceptual* articulation of thought and action," and that we have failed "to rethink from the beginning the constraints and criteria of adequacy of the enterprise in light of the expressive power the new formal idioms put at our disposal." And we must agree that the empiricist program "could not in principle do justice ... to the normativity of concept use that finds its expression variously in the distinction between laws of nature codifying inferential relations among facts, on the one hand, and mere regularities regarding them, on the other, and in the difference between acting for a reason and merely moving when prompted" [5: 32]. But can we also agree that his alternative rationalist focus, on conceptual articulation of commitments and entitlements rather than experiences and inclinations should be primary, even in scientific inquiry?

Some scholars have attempted to explain Peirce's pragmatism as a reconstructed rationalism (fundamentally deductive) or an extended or radical empiricism (fundamentally inductive). Hilda Blanco is among others who instead effectively argue that Peirce's theory responds to the dichotomy in traditional philosophy between empiricism and rationalism, which are both theories of knowledge based on visual perception (either as the primary mode of sense perception or as an internalized visual metaphor) [see 6: 55-61]. She explains that pragmatism's key concept of experience refers not to the unmediated sensory or perceptual data of the empiricists; knowing does not begin or end with experience, but is a continuous phenomenon, also not reducible to the rationalist ideal of a deductive system of ultimate indubitable propositions. Pragmatism changes the metaphor for knowing from *seeing* to *manipulating:* knowledge of things results more from the resistance or compliance things exhibit to our attempts at implementing them or to the habitual ways we respond to them, than to a passive, external perception or to some ideal insight [see 6: 45].

Peirce's pragmatism describes the means of linking perception and insight in the process of inquiry, by the guessing of possible links in abduction. Brandom contends that only against the background of insight can perceived phenomena be recognized. Peirce agrees, but stresses that when we believe we know something, we must be forced out of that rational complacency by surprising experience that we "simply ourselves with," which causes us to doubt what we believed and to proceed in inquiry toward resolving that doubt by modifying our beliefs. Knowledge requires ongoing inquiry, progressing toward the limit of what we can only *hypothesize as* "objective reality," from rationalized experience which, without abductive and inductive steps, would be only subjective and arbitrary. Peirce conceived pragmatism to account for how we can know empirical experiential relations: by inferring probable antecedents from the consequences we already know. The step of adopting an antecedent as a *hypothetical* explanation (in abduction) is the method of learning by testing ideas [see 8: 254]. Peirce first regarded pragmatism as a rule for establishing the meaning of concepts, later as a rule for selecting hypotheses, and eventually as a method for evaluating beliefs [see 8: 259-60, 147]. Finally (by 1903), Peirce extended pragmatism to cover the entire logic of abduction [CP 5.196]. In abduction, we:

1. experience some surprising or disturbing phenomenon (P),
2. suppose that P would be explicable as not surprising, if hypothetical condition (H) were true,
3. accept in these conditional terms that H might be true [see *CP* 5.189].

We can identify Peirce's logical stages of inquiry (deductive, inductive, and abductive) with Brandom's three axes of inferential relations induced by commitments, entitlements, and their interactions, defining his "normative fine structure" of rational practices (such as the production and consumption of reasons embodied in his pragmatic notion of the game). Abductive inference then corresponds to the *interactions* of commitments and entitlements (Brandom uses the term "modal," rather than "abductive") [see 5: 194; compare to Peirce's three stages of inquiry: *CP* 6.469-72]. If, in Brandom's terms, commitments are beliefs and entitlements are reasons for believing, to make explicit our implicit material inferences, then Brandom's argument for logically expressing and validating of material inference relies on Peirce's role for abductive inference. Brandom's inferentialism might be understood as trying to "harmonize" rational and empirical elements of conceptual content (in terms of commitments and entitlements), as Peirce's abduction works to direct the interaction of deduction and induction in his theory of inquiry. Helmut Pape explains: "Abduction addresses the problem of logical continuity and singularity in the dynamics of human cognition by showing that there are always non-formal contents that have to be integrated into the smooth structure of logical relations" [9: 250].

Brandom's strategy of understanding knowledge as a normative social status, as far as it goes, appears compatible with Peirce's normative science, but lacks the functional scope of Peirce's pragmatism in its operation. Reviewing that strategy: conceptual content is constituted by being committed in the sense that the game requires: "as taking up a stance in an *inferentially* articulated network—that is one in which one commitment carries with it various others as its inferential consequences and rules out others that are incompatible. Only as occupying a position in such a network can what *appears to be* a discursive expression be understood as *propositionally* (and hence *conceptually*) contentful" [5: 118]. If we take someone to know something, Brandom insists we must: "attribute a certain kind of inferentially articulated *commitment,* and attribute a certain kind of *entitlement* to that commitment." In calling something "knowledge," we are doing three things: *"attributing* a *commitment* that is capable of serving both as premise and as conclusion of inferences relating it to other commitments, *attributing entitlement* to that commitment, and *undertaking* that same commitment oneself. Doing this is adopting a complex, essentially *socially* articulated stance or position in the game of giving asking for reasons" [5: 119]. But he stresses that "not all beliefs to which the believer is entitled count as knowledge" [5: 118-19]. How then do we distinguish knowledge from beliefs? What sort of "harmony" serves as the "objective ideal" toward which pragmatism would have us strive to hone our beliefs: what should be logic's aim, according to normative science?

7 The Ultimate Challenge for Conceptual Structures Research

Missing from Brandom's strategy is what Peirce conceived his normative science to study. He maintained: "when one comes to such questions as ... the connection of mind and matter (further than that mind acts on matter not like a cause but like a law) we are left completely in the dark. The effect of pragmatism here is simply to open our minds to receiving any evidence, not to furnish evidence" [*CP* 8.259].

The value of pragmatism as the method of normative science is that "abduction commits us to nothing. It merely causes a hypothesis to be set down upon our docket of cases to be tried" [*CP* 5.602]. Hypotheses are intelligent guesses or solutions to puzzles that invite careful consideration and then perhaps testing. Conceptual Structures tools, by which we hope to *do* something (by augmenting how mind acts on matter), might be conceived as embodied hypotheses that we are prepared to test. Beyond the abductive step of tool creation, we must then *commit* ourselves to some aim (or belief) in testing (that is, to some function we *claim* for the tool), relying on logic to make explicit whatever the tool does, which will *entitle* us to that commitment. We can then be *justified* in claiming what we expect to be the tool's function, and in further testing others can *inherit* our entitlement (reason to believe the claim). Blanco suggests that in Peirce's terms, a claim would include a "fallibilistic tag" as a "reference to observations, tests, and any other evidence related to the claim, and indication of the constraints on any of these, as well as an estimate of error." This tagging of claims would indicate the source or direction of errors to aid in correcting them, to ensure "a self-correcting tendency in inquiry" [6: 65]. How such "tagging" might be used in the game (to track the interactions of commitments and entitlements) is worth investigating for use in guiding improvement.

Only by abduction do we create hypotheses (or create new solutions and tools) for puzzles we encounter in the course of inquiry [see "meta-logical" functions, in 9: 250]. Blanco stresses that logic finds its role in situations that are indeterminate, unsettled, and doubtful, because these qualities spur inquiry; while the quality of wholeness, of harmony in a situation, terminates inquiry [see 6: 66]. Peirce concludes: "all that logic warrants is a hope, and not a belief. It must be admitted, however, that such hopes play a considerable part in logic. For example, when we discuss a vexed question, we hope that there is some ascertainable truth about it, and that the discussion is not to go on forever and to no purpose" [*CP* 2.113]. That "hope" is the ultimate abduction. The purpose of Peirce's normative science is to inquire into that ultimate aim of inquiry, which drives all attempts at knowing anything more than what we immediately experience and ensures the "Non-relativity of Knowledge" [*CP* 8.112].

In Peirce's normative science of feeling, action, and thought (studied under esthetics, ethics, and logic) [see *CP* 5.130], Kelly Parker explains that esthetics describes inquiry's ultimate abduction or highest ideal as "the quality of feeling evoked by the process that evolves greater reasonableness and harmony out of the plurality of things in the universe. ... not a state of absolute harmony or absence of strife—not nirvana—but rather the *feeling that accompanies* increasing order and harmony in the world of our experience." Logic's normative aim then is "to articulate the conditions for veracity, under which thinking can reasonably be considered to increase order, harmony, and connectedness in the world of thought" [10: 32, 35]. Parker formulates what he says might be Peirce's categorical imperative: "The aims one pursues ought above all to contribute, in the long run, to the increase of order, harmony, and connectedness within one's own community and world of experience. Any action that neglects this imperative is ultimately pernicious" [10: 34; see "Gospel of Greed," *CP* 6.294]. Logic's role is to discern the regularities and laws that can be found in feeling, action, and thought alike, "the determination of such generals in fluid and chaotic experience is the key to establishing order, harmony, and connectedness in the world" [10: 35]. Logical goodness (or goodness of representation), relies on moral goodness (or veracity), which relies on esthetic goodness (or expressiveness) to advance knowledge toward truth, conceived as a unifying representation of reality in the indefinite future [see 10; *CP* 5.137-43; 5.331].

Peirce's pragmatism responds to that ultimate imperative, asking us at every step of inquiry to envision (as best we can, by whatever representational means we can) if the consequences of *how we conceive the world* would ultimately lead to that desirable future. "[Representation] is that which is what it is by virtue of imparting a quality to reactions in the future" [*CP* 1.343]. Representing knowledge, on Peirce's pragmatic account, would exemplify that evolution of inquiry, making explicit our concepts of intuition, automation, and nominalism in the normative terms of esthetics, ethics, and logic. Can conceptual structures tools be developed toward that ultimate aim through Brandom's framework?

References

General Notes: "MS" references are to Peirce's manuscripts archived at the Houghton Library, Harvard; for *CP* references, *Collected Papers of Charles Sanders Peirce,* 8 vols., ed. Arthur W. Burks, Charles Hartshorne, and Paul Weiss (Cambridge: Harvard University Press, 1931-58).

1. Keeler, M. [2003]. "Hegel in a Strange Costume: Reconsidering Normative Science in Conceptual Structures Research." In: de Moor, A., Lex, W., and Ganter, B. (Eds.): *Lecture Notes in Artificial Intelligence,* Vol. 2746, Springer-Verlag, 37-53.
2. Keeler, M. [2000]. "Pragmatically Yours," In: Ganter, B. and Mineau, G. (Eds.): *Lecture Notes in Artificial Intelligence,* Vol. 1867, Springer-Verlag, 82-99.
3. de Moor, A., Keeler, M. and Richmond, G. [2002]. "Towards a Pragmatic Web." In: Priss, U., Corbett, D. and Angelova, G. (Eds.): *Lecture Notes in Artificial Intelligence,* Vol. 2393, Springer-Verlag, 235-249.
4. Keeler, M. and Denning S. [1991]. "The Challenge of Interface Design for Communication Theory: from Interaction Metaphor to Contexts of Discovery." In *Interacting with Computers,* Vol. 3, no. 3, 283-301.
5. Brandom, R. [2000]. *Articulating Reasons: An Introduction to Inferentialism.* Cambridge, MA: Harvard University Press.
6. Blanco, H. [1994]. How to Think About Social Problems: American Pragmatism and the Idea of Planning. Greenwood Press.
7. Brandom, R. [1994]. *Making It Explicit: Reasoning, Representing, and Discursive Commitment.* Cambridge, MA: Harvard University Press.
8. Keeler, M. [forthcoming, 2004]. "The Philosophical Context of Peirce's Existential Graphs," *Cognito, Centro de Estudos do Pragmatismo Filosofia.*
9. Pape, H. [1999]. "Abduction and the Topology of Human Cognition." Transactions of the Charles S. Peirce Society, Vol. XXXV, No. 2, Spring, 248-269.
10. Parker, Kelly A. [2003]. "Reconstructing the Normative Sciences." *Cognito, Centro de Estudos do Pragmatismo Filosofia,* Vol. 4, no. 1, 27-45.

Notes

[1] Normative Science forms the heart Peirce's view of philosophy as a science, in terms of esthetics, ethics (or practics), and logic. Logic regarded as the theory of deliberate thinking implies that thinking is controlled with a view toward conforming it to a standard, purpose, or ideal; logic must be studied as a species of ethics, which is a species of esthetics. Kelly Parker explains that Peirce conceives his normative science as the study of what gives the rest of the sciences their direction: "it is in normative science that we critically examine the ends that guide our interactions with the world, including the action of knowing the world. . . . questions of value precede not only action, but they also precede most questions of fact (excepting only the most general questions of formal fact concerning mathematical relations, and those concerning the structure of experience which are raised in phenomenology)" [9: 29]. Facts are value-laden, and must be understood as such in scientific inquiry, without values being assumed to be hopelessly relative. The inquiry of Normative Science must address that problem.

[2] Thought is not necessarily connected with a brain. It appears in the work of bees, of crystals, and throughout the purely physical world; and one can no more deny

that it is really there, than that the colors, the shapes, etc., of objects are really there. Consistently adhere to that unwarrantable denial, and you will be driven to some form of idealistic nominalism akin to Fichte's. Not only is thought in the organic world, but it develops there. But as there cannot be a General without Instances embodying it, so there cannot be thought without Signs. We must here give "Sign" a very wide sense, no doubt, but not too wide a sense to come within our definition. Admitting that connected Signs must have a Quasi-mind, it may further be declared that there can be no isolated sign. Moreover, signs require at least two Quasi-minds; a Quasi-utterer and a Quasi-interpreter; and although these two are at one (i.e., are one mind) in the sign itself, they must nevertheless be distinct. In the Sign they are, so to say, welded. Accordingly, it is not merely a fact of human Psychology, but a necessity of Logic, that every logical evolution of thought should be dialogic. You may say that all this is loose talk; and I admit that, as it stands, it has a large infusion of arbitrariness. It might be filled out with argument so as to remove the greater part of this fault; but in the first place, such an expansion would require a volume -- and an uninviting one; and in the second place, what I have been saying is only to be applied to a slight determination of our system of diagrammatization, which it will only slightly affect; so that, should it be incorrect, the utmost certain effect will be a danger that our system may not represent every variety of non-human thought. [*CP* 4.551 (1906)]

3. Now it may be that logic ought to be the science of Thirdness in general. But as I have studied it, it is simply the science of what must be and ought to be true representation, so far as representation can be known without any gathering of special facts beyond our ordinary daily life. It is, in short, the philosophy of representation [*CP* 1.539 (1903)].

Improving the Testbed Development Process in Collaboratories

Aldo de Moor

Infolab, Dept. of Information Systems and Management
Tilburg University, the Netherlands
ademoor@uvt.nl

Abstract. Collaboratories are increasingly important as instruments for distributed work. They are highly complex socio-technical systems, in which often advanced ICTs need to be carefully tailored to subtle work practices and organizational structures. However, despite their importance and potential impact, not many successful examples of Collaboratories exist. One key obstacle is the complexity of the testbed development process in which the collaboratory is to evolve. In this paper, we propose a method for collaboratory improvement. We show how conceptual graph theory can be used to help improve the testbed development process.

1 Introduction

The scientific research community is one of the creators and oldest users of the Internet. Already when the first computer networks became operational at the end of the sixties, researchers started using their potential for collaboration. Although originally intended for the mere exchange of files, enthusiastic users immediately invented e-mail, one of the current killer applications. Internet-based technologies have developed at a phenomenal rate, both in reach and range. The privileged happy few from the early days have become a worldwide population of hundreds of millions of active users. Moreover, the primitive file exchange technologies from the beginning have evolved into a huge toolbox of functionalities. Much attention has been paid to the needs of individual users: e-mail, information retrieval applications like search engines, office tools, and so on. However, collaborative applications are still underdeveloped. Despite the billions of dollars poured into research into computer supported cooperative work, most collaborators still use powerful but primitive technologies such as mailing lists. Although these tools have proven to be very successful in bringing people together into virtual communities, they have many limitations such as information overload, navigation problems, primitive workflow management capabilities, and lack of customization. This negatively affects motivation and the accomplishment of joint objectives.

One major cause of problems is that the usefulness of knowledge tools is not rigorously evaluated [20]. The scientific community being global and therefore to a large extent virtual in nature, and having a strong drive to collaborate, is among the first to have become aware of the need for systematic support for the evolution of its socio-technical systems. It is no longer enough to just offer a toolbox with many hammers

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 261–274, 2004.

and nails, and then to wait for a virtual house in which to collaborate to construct itself. Instead, perspectives, methods, and techniques need to be developed in which such socio-technical systems are developed more effectively and efficiently. This takes place in collaboratories.

A collaboratory consists of "various tools and technologies ... integrated to provide an environment that enables scientists to make more efficient use of resources wherever they are located [19]". Since research collaboration is highly complex, constantly changing, and in need of many sophisticated ICTs, such integration of tools into a complete environment is not trivial. In collaboratories, a structured and fine-tuned testbed development process should therefore be adopted in which guided experiments with various technologies efficiently lead to more effective community information systems. However, what properties such an evolutionary process should have, let alone how it is itself to be supported by information systems is still unclear. In earlier work, we outlined parts of the solution for collaboratory process improvement. In [6], we described how Conceptual Graphs could be used to improve the pragmatic inquiry process needed for more focused testbed development. In [5], we explained how formally modelling tool contexts could be helpful to this purpose as well. Using this previous work as a starting point, our purpose in this paper is twofold: (1) to explicitly model the collaboratory testbed development process using conceptual graphs and (2) to use this formalization to improve this process. Conceptual graphs are the formalism of choice since they are very useful in pattern matching, which we will show to be essential in this process.

In Sect. 2, we show how collaboratories can be viewed as evolving socio-technical systems. Sect. 3 studies some knowledge structures useful for the modelling of collaboratory improvement. In Sect. 4, we present our method for collaboratory improvement. We end the paper with conclusions.

2 Collaboratories: Evolving Socio-technical Systems

Already from the beginning, collaboratory research has focused on socio-technical systems. For example, the definition adopted by the National Science Foundation in the early years of collaboratory research was: " [A collaboratory consists of] various tools and technologies [...] integrated to provide an environment that enables scientists to make more efficient use of resources wherever they are located" [19]. However, the insight is gaining ground that collaboratories are especially entities that support rich and recurring interaction around a common research focus, the critical element for their success thus being the opportunities they allow for encounters, discussions, and the sharing of ideas [10]. A very important question therefore is "how to support communication that permits human cooperation even when the evolutionary social mechanisms that depend on proximity are absent [15]."

2.1 The Testbed Development Process

Despite their great potential, collaboratories have not grown as much as originally envisioned. This is not so much a failure of the original vision, but a consequence of the great difficulty of supporting complex group work in virtual settings [10]. Successful collaboratory development requires (1) a system architecture and integration to explore ways that people and machines can use component technologies most effectively, (2) a research program to study the conditions required for collaboration, and (3) user-oriented rapid-prototyping testbeds, to understand the impact of the technologies used [19]. Already in the very first report on collaboratories, the essential role of testbed development for collaboratory construction in various scientific disciplines was stressed [17]. Testbeds must be established in actual working contexts, where the required tools for data access and communication can be designed, developed, and tested within a program of prototyping, testing, and evaluation that can support continuous development. Mary Keeler's PORT (**P**eirce **O**nline **R**esource **T**estbeds) project is an important initiative, which aims to produce an integrated context view of such development, using a collaboratory on the interpretation of Charles Peirce's manuscripts as an example [13].

To conceptualize an approach for formally modelling and supporting the testbed improvement process, we build on two streams of thought: Douglas Engelbart's ideas on improving the process of co-evolution of the social and the technical systems, and socio-technical interaction networks as a way of formally modelling improvement patterns.

Engelbart has spent his life studying how to improve systems development by long-term, pragmatically guided whole-system evolution. With today's dazzling rate of technological progress, tool functionalities are developing faster than man's ability to use them to the fullest. Engelbart's mission is to find ways to accelerate our intellectual development, so that we can keep up with our tools and improve their co-evolution with work practices [12]. In his view, one is notjust to focus on the co-evolution of Human and Tool-systems, but also to continuously improve the design process itself [8]. In his CODIAK (**CO**ncurrent **D**evelopment, **I**ntegration and **A**pplication of **K**nowledge) process, he describes a vision for increasing the capabilities of organizations to "improve their improvement" [9]. It consists of intelligence collection (to identify problems, needs, and opportunities), dialog records (to conduct and coordinate improvement dialogs), and knowledge products (that capture relevant improvement project status and plans). However, these artefacts are not an end, but only a means to improve improvement capabilities and processes. This is in line with modern views on knowledge management, which claim that tacit and explicit knowledge both have specific, inter-related roles to play in optimizing knowledge creation processes [18].

A good way to capture the structure and behaviour of collaboratories, is to see them through the lens of socio-technical interaction network (STIN) theory, which is rooted in actor-network theory [14]. Using the theory to do careful empirical analyses of work situations, it was found that almost identical technologies are often configured very differently in practice. The theory acknowledges that collaboratories can be seen as layered systems of technical and social components. However, it takes issue with the fact that in traditional socio-technical systems analysis the technology often predominates. In contrast, STIN analysis has a more integrated view of the interaction between humans and technologies. Technologies are not a given, to which the social system has to adapt. Rather, there are many social characteristics that help shape the technologies, but are also being formed by them. STIN models take into account such issues as actor relations, content control, resource dependencies, work to make the system useful and sustainable, translations to mobilize resources, and business model and governance structures. The models can be used to analyze the complex interactions between people, between people and technologies, and between technologies themselves. Social analysis is required in all stages of the collaboratory lifecycle, including planning, development, configuration, use, and evolution. These forms of analysis acknowledge that 'users' are too shallow a construct for obtaining useful insight in collaboratory improvement. Instead, the models provide socially richer characterizations of people working and communicating in complex, overlapping socio-technical networks.

Combining Engelbart's views on testbed development improvement with the basic ideas behind socio-technical interaction networks seems a feasible approach toward modelling testbed development improvement. However, both approaches are still qualitative, informal and disjoint. Collaboratory improvement, with all its inherent operational and evolutionary complexities, could benefit greatly from a - partially - formal approach. In this way, more systematic analysis methods and supporting tools can be developed. In the remainder of this article, we will outline the foundations of how such an approach could look like, using conceptual graphs as the knowledge representation and reasoning formalism. First, we introduce some relevant knowledge structures.

3 Knowledge Structures for Collaboratory Improvement

Based on Engelbart's views, we distinguish several systems providing contexts of interpretation of tools. Next, we acknowledge the need for improvement patterns, and analyze one promising direction: a socio-technical pattern language. We then explain why conceptual graphs are our knowledge formalism of choice.

3.1 Tool Contexts in Testbed Development

As a starting point for our approach, we use Engelbart's insights into co-evolution improvement to create a layered tool context model. Focal constructs in these contexts are *processes,* since obtaining better quality testbed dynamics is at the core of what is needed. Based on Engelbart, we distinguish four layers of testbed processes:

- *Information/Communication (I/C) processes*: the processes enabled by the tools.
- *Workflows:* the processes in which tasks are executed and coordinated so that the goals of the community can be accomplished.
- *Design processes:* the processes in which the users reflect on their work and I/C processes and propose modifications to the design of their socio-technical system so that workflows can become more effective and efficient.
- *Improvement processes:* the processes in which the design processes themselves are made more effective and efficient.

These processes are organized in four nested systems: the information system, work system, design system, and improvement system respectively. Each higher-order system provides a context for the system it embeds. For example, the information system is embedded by the work system, which thus defines the context of *use* of the information system. Similarly, the design system provides the context of *change* of the work system, and the improvement system the context of *optimization* of the design system. Collaboratory improvement can only occur systematically if all systems and their interfaces are clearly defined at the level of detail required by the particular community. The advantage of modelling collaboratory improvement as a set of embedded systems is that it allows for the abstracting of details less relevant to a particular improvement purpose. For instance, sometimes the improvement may focus on the information system, when a new tool needs to be evaluated. In other cases, the community may not be interested so much in the particular details of its current socio-technical system, but want to focus on how it can improve its evolutionary practices. Using this contextual framework, testbed development methods can be more systematically created, interpreted, and changed.

3.2 Improvement Patterns

Using the tool context model allows us to conceptually distinguish between the layered elements of improvement knowledge. However, in order to reason about properties of this knowledge, such as about *knowledge gaps*, we need to use modelling approaches that are well suited to the particulars of this kind of knowledge. As the theory and findings on socio-technical interaction networks show, socio-technical systems knowledge evolves in *fragmented*, partial ways. Furthermore, testbed development knowledge often has different degrees of *specificity*: at the beginning of a project, some aspects, such as which particular workflows to support by what tools can remain undefined, while some other issues, as who to involve in a particular design process may need to be very precisely defined. How to combine this fragmented and diffuse knowledge evolution process with the systemic view proposed by Engelbart?

Patterns are good ways to capture such knowledge and place it in a system context. A pattern is something designed or used as a model for making things (Merriam-Webster). The following statements more precisely indicate the power of patterns:

> A pattern is a careful description of a perennial solution to a recurring problem within a building context, describing one of the configurations which brings life to a building.
>
> A pattern language is a network of patterns that call upon one another. Patterns help us remember insights and knowledge about design and can be used in combination to create solutions. (both quotes from Alexander et al, 1977, in [22]).

Humans are very good at using patterns to order the world and make sense of things in complex situations [16]. In the information systems literature, patterns are gaining prominence as a way to deal with the complexity and dynamics of the real world. For example, workflow patterns can be used to develop ideas on how to implement certain business requirements given that a workflow server already exists. Others focus on developing pattern languages that capture communication knowledge of large, distributed communities [22]. In collaboratory evolution, a structured representation facility for efficiently reporting, tracking, and mapping advancements within projects is essential, making it possible to compare patterns of development and trace similarities and differences among project technical requirements [13].

Thomas et al. give a good description of the role that patterns can play in socio-technical systems improvement, and propose a socio-technical pattern language to make the software development cycle more effective and efficient [24]. In outlining the elements and use of this language, they pose three important questions: (1) how to guide pattern authors to help produce clear, understandable and helpful patterns? (2) how to support users to explore a design space using patterns? (3) what is the relation between a pattern and software components?

Important as an informal pattern language may be, it is not enough to ensure effective and efficient pattern use. Improvement patterns outline dependencies between events in the collaboratory and actions to be taken, by the system or the users. Often, human beings will need to be the agents to signal the need for change, but computers can help monitoring and managing the complex chain of change actions required. Even in small examples like the case mentioned next, people are already at a loss of doing so, as many of the case participants have reported. Thus, the informal pattern language helps to identify what needs to be done when, but in addition some (semi)-formal knowledge representation and reasoning method can be helpful to activate that knowledge. For this purpose, conceptual graphs are our formalism of choice.

3.3 Using Conceptual Graphs

Collaboratories are complex and dynamic networks. Important properties of networked societies are that they are loosely organized, their boundaries are permeable, they are often imperfectly integrated, have (reconfigurable) nodes that may be part of other networks, and have flat and recursive hierarchies [14,25]. Semantic networks are useful forms of network knowledge representation, as they use links to both record facts and to provide associative access paths, by which facts can be accessed from each other. These paths are then the basis for efficient reasoning algorithms [27]. In fact, the ability to represent and use these links is essential in defining the knowledge of a collaborative community [2].

Conceptual graphs are a flexible and extensible method for knowledge representation. Conceptual graphs are particularly useful forms of semantic networks, as they also include generalization hierarchies (of types, relations, and complete graphs), with a set of powerful operations that make use of properties of the hierarchies of graphs and their components. Conceptual graphs not only allow the description of complex domain knowledge, but also to validate that knowledge against meta-knowledge about the domain. Thus, they are very well suited to represent and reason about the context of pattern knowledge, which, as we argue, is a prerequisite in collaboratory modelling.

4 Towards a Method for Collaboratory Improvement

The previous section outlined the main knowledge structures needed to conceptualize collaboratory improvement, and presented a context- and pattern-based view on the testbed development process. The current section describes the method that formalizes the *process* in which these knowledge structures are put to use. The method uses conceptual graphs as a formal backbone to initialize focused conversations for specification.

To illustrate the method, we first introduce a case. We then examine the role of conversations for specification in collaboratories. To better support these conversations, a collaboratory improvement ontology is introduced, which forms the conceptual basis for our architecture of collaboratory improvement systems.

4.1 Case: The Tools-Yak Community

Blue Oxen Associates[1] aims to develop the art and science of collaboratories and in this way to contribute to the common good. One of its initiatives is to develop collaboratories on collaboratories. To this purpose, it hosts several mailing lists, one of them being the Tools-Yak list[2]. By discussing the ways tools could and should be used in collaboratories, and then implementing and testing proposed solutions, the community formed around this list aims to find principles and practices that contribute to better socio-technical systems for collaboratories. The list has been operational since December 2002. The list has attracted and sustained high-quality discussions, resulting in several interesting experiments and evaluations. As such, it seems a good candidate to illustrate some of the ideas proposed in this paper.

To get an idea of the complexity of collaboratory conversations, we show some illustrative data. For the example, we analyzed the mailing list archive of the first half year of the Tools-Yak community. As in this paper we cannot go into depth, we give an example of only one particular tool being examined in the community: Purple Numbers. One problem with Web pages is that it is hard to refer to specific items in a page. This makes linking often too coarse-grained. To increase the granularity of web links, software has been developed to automatically add 'purple numbers' to each paragraph in a web page. Instead ofjust linking to a page and then making cumbersome and possibly erroneous references to the "second section, first line", one can now include a link to the specific paragraph of a Web document. It is a simple (technical) idea, but turns out to have important socio-technical ramifications. This is illustrated by the following key indicators:

From December 2002 - May 2003, 95 mails were exchanged on the implementation or use of purple numbers. 17 people were involved in those discussions, with an average of 5.6 mails per person, and a standard deviation of 6.0. In this period, purple numbers were mentioned in no less than 23 threads. The average length of those parts of the threads in which purple numbers were mentioned, was 4.1, with a standard deviation of 6.6. There were several interesting outliers: one person, a coordinator of the community, contributed 24 messages. Furthermore, one thread, on the role of purple numbers in e-mail was extremely long: 31 messages.

Quantitative data on collaboratory improvement are still scarce in the literature. Although benchmarks are lacking to fully interpret their meaning, the presented indicators may help to appreciate the complexity of collaboratory development. They are typical of successful virtual communities, in which there is broad participation, some of it initiated by a facilitator, and in which a wide range of topics is discussed, sometimes leading to passionate debate. However, in Tools-Yak, many community members have voiced

[1] http://www.blueoxen.org

[2] http://collab.blueoxen.net/forums/tools-yak/

the concern that the quality of the collaboratory improvement *content* is excellent, but that many good ideas are lost because of their fragmentation across threads and lack of follow-up. Thus, much is to be gained by improving the quality of the collaboratory improvement *process*.

4.2 Conversations for Specification

The lifeblood of any community, collaboratories not excluded, are conversations. There are many types of work-related conversations, one of which is the conversation for action, in which the goal is to coordinate explicit cooperative action [26]. Another major class of conversations in collaboratories are *conversations for specification*. We define such a conversation as a self-contained unit of communication to accomplish certain specification objectives, like the specification of an experiment for testing a tool in its context of use. Evidence for the effectiveness of predefined conversation models is ambiguous, however [1]. We therefore require a conversation to be only partially structured in the sense that only main specification process entities need to be defined. However, the format of the utterance acts in which these definitions are made can remain free, as is the case in e-mail conversations. We propose that such partially-formalized conversations are crucial in providing adequate support for collaboratory improvement.

There is great value in free-form e-mail message exchange, as it does not constrain users in artificial formats which may have little meaning to them. However, e-mail does not support productive conversations per se [21]. Its free form is not only its strength, but also its weakness. Most explicit dependencies in e-mail are chronological: somebody sends an e-mail with a question or an idea, one or more people respond, those replies themselves attract new replies, and so on. Mostly, e-mail discussions on a particular topic are prolonged, divergent, and repetitive. Such problems lead to many process inefficiencies, as has been studied extensively in, for example, the literature on the IBIS (issue-based information systems) paradigm [3]. We conjecture that such conversation process inefficiencies, not lack of motivation, may be one of the most important reasons that successful collaboratories are so few and far between.

The question now is: can we reduce conversation inefficiencies using some form of (conceptual graph-based) formalization, without losing the strengths of informal (e-mail-based) conversation? In other words, can formalization, used wisely, contribute to collaboratory improvement? The answer is to be found in defining a practical form of *incremental formalization*. This enables users to choose when and how to add finer-grained, computer-readable codification to informal content [23]. In this process, boundary objects are defined: objects which are both plastic enough to adapt to local needs and the constraints of several parties employing them, yet robust enough to to maintain a common identity across sites (Star and Greisemer in [23]). These formal boundary objects form a process space that *circum*scribes rather than exactly prescribes all aspects of collaborative work [11]. As such, this way of thinking fits very well with the pattern-oriented design philosophy.

4.3 A Collaboratory Improvement Ontology

The core of the formal part of our architecture is a collaboratory improvement ontology. The current ontology is by no means complete, but forms a sufficient root hierarchy to be further refined and extended in future collaboratory research efforts.

At the heart of collaboratory improvement is the process of pragmatic inquiry, as described in [6]. Here, the improvement process is seen as a continuous process of hypothesis testing on the role that tools should play in the collaboratory. Proposed hypotheses on, for example, which tools to use to support a particular workflow, can be implemented and tested, and their usage evaluated based on certain community-defined criteria, such as security and userfriendliness. After evaluation, a hypothesis is labelled either as failed or succeeded. The socio-technical system itself can be defined in many different constructs, three of which are named here: *element*-definitions describe parts of the various (information, work, design, and improvement) subsystems making up the total socio-technical system. *Mappings* connect elements from these subsystems. One example of a mapping is a support definition, which links tools in the information system to workflow definitions in the work system. *Socio-technical (system) patterns*, finally, can be constructed of any (cross-sections) of the other elements. For example, “Who Speaks for Wolf” is a powerful socio-technical pattern aimed at engaging all the stakeholders in a design discussion, also those who are absent at a meeting[24]. The focus of this particular pattern is on roles and processes in the design system: make sure to involve all end users in changes to their socio-technical system, etc. However, it also optimizes this process in the improvement system, specifying, for example that *bad* designs need to be weeded out, that stakeholders should be involved in the design process *early*, etc. Such *qualifications* of the design process should be modelled in the highest-level improvement system. The type hierarchy of the ontology is given next.

```
T >                      T >                  T >
 Criterion >            Process >            System >
  Definition >            Design_Proc          Des_Sys
   Element                I/C_Proc             Impr_Sys
   Mapping >              Impr_Proc >          Info_Sys
      Support               Propose_Hyp        Work_Sys
   STS_Pattern              Test_Hyp         Tool >
 Hypothesis >             Workflow >           Mailing_List
   Prop_Hyp                 Archive            Wiki
   Tested_Hyp >             Discuss            ...
     Failed_Hyp             ...
     Succ_Hyp
```

Note that many of these concepts have proper type definition graphs. These can be used to enforce required semantics, but have been omitted for lack of space. Furthermore, this ontology contains some domain-specific concept subtypes, such as Mailing List and Wiki-Tools, and Discuss and Archive-Workflows. In other cases, these specific types could be different.

4.4 An Architecture for Collaboratory Improvement Systems

Fig. 1 outlines the architecture of the system we propose to be used in support of the collaboratory improvement process. It is based on Dan Corbett’s important vision of

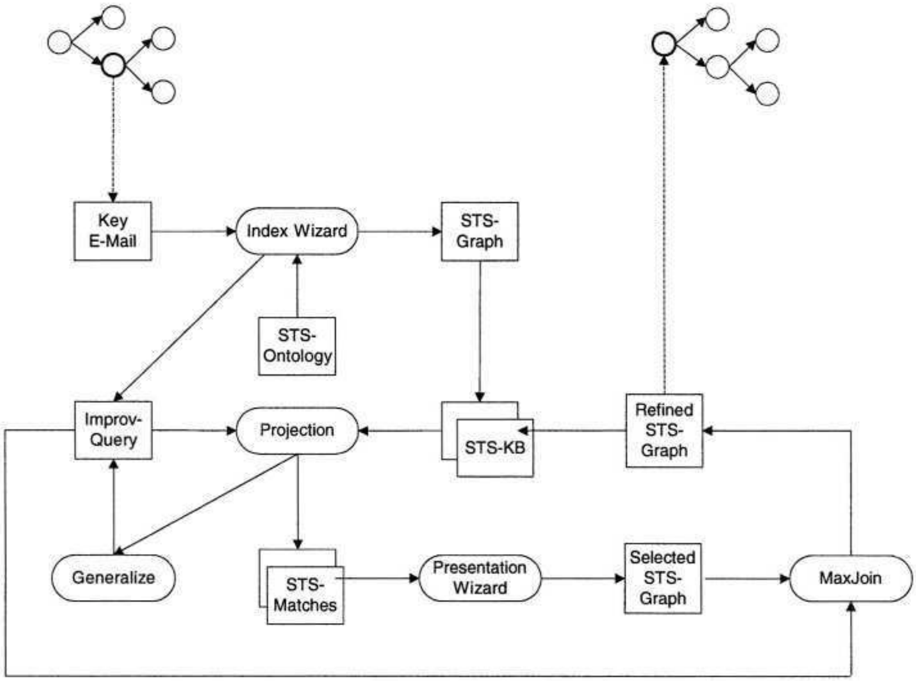


Fig. 1. An Architecture for Collaboratory Improvement Support

at unification can be made [4]. The outline of our architecture is as follows. As e-mail conversations for specification take place, users can identify *key mails.* These are e-mails in which, in the user's opinion, important collaboratory improvement suggestions are made. For example, the author of the very first e-mail to Tools-Yak, Eugene, indicated that two tools had been installed: Mailing Lists and Wikis. The mailing list was to support the discussion and list archiving-workflows, whereas the Wiki was to be used for management of the knowledge obtained in the collaboratory discussion.

As soon as a key mail has been recognized, the Index Wizard is invoked. In a simple (pseudo-natural language) dialogue, the user is quizzed by the system. To do so, it can use (an extended version of) the ontology given above. Many advanced querying techniques have been developed in the CG community over the years. A very simple scenario of how a key mail could be indexed is the following.

- After Eugene indicates that the current mail is a key mail, the system presents Eugene with list of indexing options:
  (a) Describe tool functionalities;
  (b) Describe workflow properties;
  (c) Describe workflow support
- Eugene selects option (c)
- The system retrieves all subtypes of this Tool-concept, including Blog, Mailing_List and Wiki; the same goes for the subtypes of Workflow. The system presents these

concepts as a simple HTML-pulldown menu, from which Eugene has to select which workflow is supported by which tool. Furthermore, he is asked whether the discussion is about the results of an experiment already conducted, and whether this experiment was successful or not, or whether it is about a new experiment. The system would store this result by adding a relation to the STS-graph with a Succ_Hyp, Failed_Hyp, or Prop_Hyp concept, respectively.

The result of this human-machine dialogue is an STS (socio-technical system)-graph, which is stored in the STS-knowledge base. In case of the first mail, this graph obtained from the dialogue could look like this:

```
[STS_Pattern] -
   (Part) -> [Prop_Hyp]
   (Part) -> [Support] -
                (Inst) -> [Mailing_List]
                (Obj) -> [Discuss]
   (Part) -> [Support] -
                (Inst) -> [Mailing_List]
                (Obj) -> [Archive]
   (Part) -> [Support] -
                (Inst) -> [Wiki]
                (Obj) -> [Knowledge_Management]
```

Note that the graph represents a proposed hypothesis: the coordinator *expects* the tools to be used this way, but as the intensity of the discussion in the following half year has shown, many modifications were proposed. Had our system been available, many additional graphs could thus have been added to the knowledge base.

Now, assume that Mary, the coordinator of the PORT collaboratory, would like to find out about possibly useful experiences and contacts related to the following question she has: are there actually any successful user experiences with tools for knowledge management in collaboratories? Her interaction with the index-wizard leads to the following formal representation of her query:

```
[STS_Pattern] -
   (Part) -> [Succ_Hyp]
   (Part) -> [Support] -
                (Inst) -> [Tool]
                (Obj) -> [Knowledge_Management]
```

To find relevant conversations, the improvement query is projected on the STS-KB. In this case, the STS graph representing Eugene's mail is not retrieved, as it is not a specialization of the query. Assuming this is the only graph in the KB, the query fails. In a subsequent dialogue with the wizard, Mary decides to generalize her query: she is not just interested in successful experiments, but in *any* experiment. The first Part-relation of her query is therefore dropped. Querying the knowledge base again this time returns Eugene's graph. The Presentation Wizard uses this graph to initiate a dialogue with Mary on whether this is what she wants. The conversation looks interesting, and she wants to know more. The system then creates a maximal join of her query with Eugene's graph, stores this new graph in the KB, and initiates a new conversation to which both Mary and Eugene are invited. The link to the initial node of this conversation is stored with the graph in the knowledge base. The participants can then use the normal discussion tools like e-mail and mailing lists to exchange tips and tricks. If in the future somebody has a query like Mary's initial one, this person will now immediately be guided to the log of the new query, which is much more specific than the thread started by Eugene's initial mail.

The system has been partially implemented using Adil Kabbaj's PROLOG+CG[3]. Of course, many technical extensions are conceivable. Once pointed to relevant discussions, the users could also use the more traditional discussion tool navigation functionality, for example search-options in a mailing list-archive, to further expand the context of interpretation of a particular improvement proposals. Additionally, content-based automatic mail-indexing tools, such as developed in the FCA community could also be added. However, such enriched content analysis is not the focus of this paper: our contribution has been to find a subtle balance between the strength of human collaborators (interpretation of rich e-mail content) and the power of machine systems (automatic inferencing of complex pattern knowledge), so that current bariers to socio-technical system evolution can be reduced.

In sum, the contribution of this approach is that collaboratory improvement is framed as a problem of socio-technical system evolution; that these abstract ideas are operationalized in a practical use-situation ; and that a fine-balanced mix of human natural language interaction is coupled with powerful graph matching to find links to rich human conversations, to be interpreted by people. The innovation of this approach is that the indexing by people is very simple, and hardly disruptive, while simultaneously being semantically very rich, as the index graphs are framed in terms of a socio-technical system improvement ontology. This meta-level reasoning, which makes good use of generalization hierarchies of the index graphs, helps people find relevant mails more easily than possible with current keyword searches. Such an approach will become truly powerful when multiple collaboratories start using this or a similar ontology: cross-community learning can then take place between communities that do not even know each other.

5 Conclusions

Improvement patterns are an essential element of collaboratory evolution. They capture collective wisdom and can be used in various ways, for example, in guiding discussions or designing tools that are customized to the complex needs of a particular community. Although improvement pattern *content* is quickly maturing, many *process* inefficiencies of how to effectively use these many rich and - by definition - partial patterns remain.

In this paper, we proposed the outline of a semi-formal method to help collaboratories more efficiently establish contacts and tune in to relevant informal collaboratory improvement discussions, across cases and communities. Our model is grounded in Engelbart's context-based philosophy of the socio-technical system improvement process in combination with a pattern-based view to deal with the partiality and specificity of this process. We formalized this model using conceptual graphs.

The basis of the formal model is an ontology of collaboratory improvement. We showed how the collaboratory improvement process can be modelled as a search for a match between an improvement query graph and the graph definitions representing the

[3] http://www.insea.ac.ma/CGTools/PROLOG+CG.htm

existing socio-technical system. Queries themselves become the basis for new socio-technical system graphs. All graphs are linked to specific conversations in the collaboratory. The main use of the retrieved graphs is to act as indices to previous community discussion, and as initiators of new discussion.

The ultimate goal in collaboratory improvement is to build active knowledge systems: systems that have the capability to solve practical and complex problems. Crucial to such systems is that they can inter*act* and not just inter*face* with the real world [7]. A collaboratory improvement system that would make use of the approach described in this paper, combined with, for instance, actor-based checks for improvement opportunities and automatic notification of key users, could be a major step forward on the way to more powerful and tailored collaboratory development.

Another implication of this work, is that it may give conceptual graph theory a class of practical and theoretical problems that do justice to its power and elegance. It has often been said that this theory was ahead of its time, in a way a solution looking for a matching problem. The area of collaboratory improvement, with its high significance in for instance the research and business domains could prove to be one of the 'killer problems' our field has been waiting for.

By combining the representational and reasoning power of conceptual graphs with the - fortunately - unique capabilities of human beings to interpret and frame improvement problems and their solutions, we have presented one operationalization of the essence of Engelbart's vision on using computer technology to *augment* collaborative communities. We hope that others will extend this preliminary work and use it to develop the collaborative methods and systems that are essential in an ever more complex and dynamic society.

References

1. E. Auramäki and K. Lyytinen. On the success of speech acts and negotiating commitments. In *Proceedings of the First International Workshop on Communication Modelling, the Language/Action Perspective (LAP'96), Oisterwijk, The Netherlands, July 1-2, 1996,* pages 1–12, 1996.
2. M. Bieber et al. Towards knowledge-sharing and learning in virtual professional communities. In *Proc. of the 35th Hawaii International Conference on System Sciences, Hawaii, January 5-7,* 2002.
3. J. Conklin, A. Selvin, S. Buckingham Shum, and M. Sierhuis. Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In *Proc. of the 8th International Working Conference on the Language/Action Perspective on Communication Modelling, Tilburg, the Netherlands, July 1-2,* 2003.
4. D. Corbett. *Reasoning and Unification over Conceptual Graphs.* Kluwer Academic, New York, 2003.
5. A. de Moor. Making Doug's dream come true: Collaboratories in context. In *Proc. of the PORT Pragmatic Web Workshop, Borovets, Bulgaria, July 15,* 2002.
6. A. de Moor, M. Keeler, and G. Richmond. Towards a pragmatic web. In *Proc. of the 10th International Conference on Conceptual Structures, (ICCS 2002), Borovets, Bulgaria, July 15-19,* Lecture Notes in Artificial Intelligence. Springer-Verlag, 2002.
7. H. Delugach. Towards building active knowledge systems with conceptual graphs. In *Proc. of the 11th International Conference on Conceptual Structures (ICCS 2003), Dresden, July 2003,* pages 296–308, 2003.

8. D. Engelbart. Coordinated information services for a discipline- or mission-oriented community. In *Proc. of the 2nd Annual Computer Communications Conference, San Jose, California, January 24,* 1973.
9. D. Engelbart. Toward high-performance organizations: A strategic role for groupware. Technical report, Bootstrap Institute, 1992.
10. T.A. Finholt. Collaboratories as a new form of scientific organization. *Economics of Innovation and New Technology,* 12(1):5–25, 2003.
11. G. Fitzpatrick and J. Welsh. Process support: Inflexible imposition or chaotic composition? *Interacting with Computers,* 7(2):167–180, 1995.
12. J. Gillies and R. Cailliau. *How the Web Was Born.* Oxford University Press, 2000.
13. M. Keeler. Collaboratories: Improving theory and method. In *Workshop on Innovations in Digital Asset Management, Fraunhofer/IPSI, Darmstadt, Germany, October 6-8,* 2003.
14. R. Kling, G. McKim, J. Fortuna, and A. King. Scientific Collaboratories as socio-technical interaction networks: A theoretical approach. In *Proceedings of AMCIS 2000, August 10-13, Long Beach, CA,* 2000.
15. R.T. Kouzes, J.D. Myers, and W. Wulf. Collaboratories: Doing science on the Internet. *IEEE Computer,* 29(8):40–46, 1996.
16. C.F. Kurtz and D.J. Snowden. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systems Journal,* 42(3):462–483, 2003.
17. J. Lederberg and K. Uncapher. Towards a national collaboratory: Report of an invitational workshop at the Rockefeller University, New York City, march 17-18. Technical report, National Science Foundation, 1989.
18. I. Nonaka, R. Toyama, and N. Konno. SECI, ba and leadership: A unified model of dynamic knowledge creation. *Long Range Planning,* 33:5–34, 2000.
19. NRC. National collaboratories: Applying information technology for scientific research. Technical report, National Research Council, Committee Toward a National Collaboratory: Establishing the User-Developer Partnership, Washington, D.C., 1993.
20. T. Renkema and E. Berghout. Methodologies for information system investment evaluation at the proposal stage: A comparative view. *Information and Software Technology,* 39(1):1–13, 1997.
21. D. Sanderson. Collaborative and cooperative mediated research. In T.M. Harrison and T. Stephen, editors, *Computer Networking and Scholarly Communication in the Twenty-First Century University,* pages 95–114. State University of New York Press, 1996.
22. D. Schuler. A pattern language for living communication. In *Participatory Design Conference (PDC'02), Malmo, Sweden, June,* 2002.
23. S.B. Shum and A.M. Selvin. Structuring discourse for collective interpretation. In *Proc. of Distributed Collective Practices 2000: Conference on Collective Cognition and Memory Practices, Paris, September 19-20,* 2000.
24. J. Thomas, C. Danis, and S. Greene. Socio-technical pattern language proposal. In *Pattern Language Workshop,* 2002.
25. B. Wellman. Computer networks as social networks. *Science,* 293:2031–2034, 2001.
26. T. Winograd. A language/action perspective on the design of cooperative work, report no.CSLI-87-98. Technical report, Center for the Study of Language and Information, Stanford University, May 1987.
27. W.A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge,* pages 45–95. Morgan Kaufmann, San Mateo, CA, 1991.

Relation Elements for the Semantic Web

Cartik R. Kothari and David J. Russomanno

Department of Electrical and Computer Engineering
The University of Memphis
Memphis, TN 38152 USA
{rkothari, d-russomanno}@memphis.edu

Abstract. The investigation into the intrinsic nature of relations has led to the development of the Relation Element theory. At present, there is no significant initiative to comprehensively describe the intrinsic nature of relations in ontological representations for the Semantic Web. The use of relation elements can describe the nature of relations in knowledge domains beyond the capabilities of currently popular knowledge representation paradigms for the Semantic Web such as OWL and DAML. With this in mind, this paper presents an implementation of the relation element theory for the Semantic Web using the reification feature of RDF/RDFS. The reasons for the inability to use DAML or OWL for defining relation elements are discussed with possible solutions. Finally, some of the possible benefits of using relation elements to define the intrinsic nature of relations for Semantic Web applications are also proposed and demonstrated in this paper.

1 Introduction

The chief objective of the Semantic Web [1] is to format the extensive knowledge, data and meta-data stored in the World Wide Web for processing by intelligent agents. This involves using standardized sets of markup tags on Web pages that can be used to encode the information contained in them.

In keeping with the decentralized nature of the Web, the Semantic Web is a collection of ontological representations of knowledge domains spanning the dimensions of human knowledge. The ontological representation of a knowledge domain comprises definitions of classes, individuals and the properties of those classes and individuals. Properties capture the descriptive attributes of classes and individuals; and also the relationships between them.

The Resource Description Framework (RDF) [2] and RDF Schema (RDFS) [3] were among the very first markup formalisms that could be used on Web pages to encode information in the context of the Semantic Web. RDF can be used to declare the meta-properties of Web pages or "resources" in the form of Object-Attribute-Value triples while RDFS provides a standard set of constructs used by RDF representations at a higher level of abstraction. In summary, RDF/RDFS is a very basic Knowledge Representation (KR) formalism that can be used to define classes and properties and to specify subsumption hierarchies among the defined classes and properties.

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 275–286, 2004.

The main drawback of RDF/RDFS is the lack of formal underlying model semantics as pointed out by Pan and Horrocks [4]. This drawback is absent in more recent and popular Semantic Web languages such as the DARPA Agent Markup Language (DAML) [5], developed by the Defense Advanced Research Projects Agency (DARPA), the Ontology Inference Layer (OIL) [6] and the Web Ontology Language (OWL) [7, 8], developed by the World Wide Web Consortium (W3C). These languages are based upon Description Logic (DL) formalisms [9], which provide very rich semantics for the definition of concepts and their relationships in a knowledge base as compared to RDF/RDFS.

However, the increased semantic expressiveness offered by these underlying DL systems is more biased towards classes (or concepts) than relations (or roles) between these classes. It is possible in these DL systems to make statements about sets of concepts such as to declaratively specify that two classes are disjoint. However, once a fixed conceptual framework is established, that is, the classes and relations have been defined for some universe of discourse, analogous declarative statements are not possible about sets of relations due to the absence of reification mechanisms. This DL system limitation has been noted in our earlier work regarding the modeling of UML inter-link constraints asserted upon associations using OWL [10]. For example, it is not possible in DL systems to assert that two relations are disjoint, nor is it possible to make statements about the intrinsic nature of relations. It is proposed that augmenting DL systems and DL-based languages, such as OWL and DAML, with richer semantics for relations, including mechanisms for reification, will benefit inference mechanisms on the Semantic Web.

The next section presents an effort using RDF/RDFS to represent the intrinsic nature of relations beyond the capabilities of DL-based languages. The reasons for the impossibility of defining relation elements using DL-based languages such as OWL and DAML are also discussed. Section 3 presents an overview of relation element theory and discusses in detail some of the possible ways in which richer relation semantics can benefit knowledge representation and management for applications on the Semantic Web. Section 4 outlines conclusions and possible future areas of work.

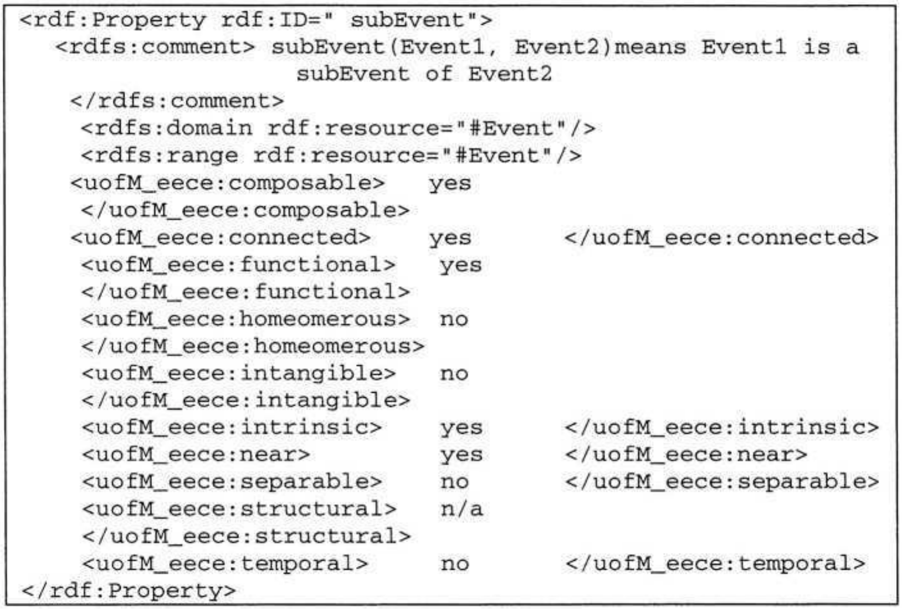
2 Using RDF/RDFS for Relation Semantics

Huhns and Stephens [11] have defined a set of meta-properties or "relation elements" viz. *composable, connected, functional, homeomerous, intangible, intrinsic, near, separable, structural* and *temporal* that can be used to describe the intrinsic nature of a relation between two entities. Given a relation, each of these meta-properties can take a value from {*yes, no, n/a*} to denote, *yes* the property holds, or *n/a* the property is not applicable or *no* the property does not hold between the domain and range elements of the specified relation. This vector of meta-properties can be used to describe 3^{10} different relations on the basis of the nature of the relationship between the domain and range. Additional relations could be defined when accounting for the possibility that two given relations could potentially have the same vector instantiation, but differ in their domain and/or range; therefore making each unique. The definition of each of these relation elements is as follows [11]:

a) Functional: If this relation element takes the value *yes,* the domain element is in a specific position that supports the functional role of the range element.
b) Homeomerous: If this relation element takes the value *yes,* the domain element is made of the same “stuff as the range element. Moreover, every domain element paired with a given range element must be made of the same “stuff”.
c) Near: If this relation element takes the value *yes,* the domain element is spatially or temporally near the range element. In terms of time, the domain and range elements would be contemporaneous while in terms of space, they would be located in physical proximity to one another.
d) Separable: If this relation element takes the value *yes,* the domain element can be separated from and can exist independently of the range element
e) Composable: If this relation element takes a value of *yes,* the relation can be composed with other relations.
f) Connected: If this relation element takes the value *yes*, the relation element is temporally/physically connected to the range element directly or transitively.
g) Intangible: If this relation element takes the value *yes,* the domain and the range element have a hierarchical relationship with regard to ownership or mental inclusion.
h) Intrinsic: If this relation element takes the value *yes,* the relation can be viewed as a descriptive attribute of the domain element. If it takes a value *no,* the relation is simply a descriptive attribute of the range element.
i) Structural: If this relation element takes the value *yes,* the domain and range element have a hierarchical relationship with the domain element below the range element in the hierarchy. If it takes a value *no,* the range element is below the domain element in the hierarchy.
j) Temporal: If this relation element takes the value *yes,* the domain and the range elements are temporally related to one another with the domain element preceding the range element. If it takes a value *no,* the range element precedes the domain element.

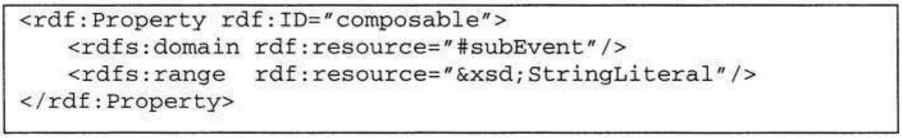
In [12], the reification feature of RDF/RDFS has been used to implement a vector of relation elements to describe every relation in a simple ontology. Reification is the most controversial and least understood feature of RDF/RDFS. Reification allows for statements about classes and properties to be treated as classes themselves. The controversy arises because reification allows knowledge engineers to avoid asserting the validity of the information or knowledge that they provide. Used in a restricted sense however, it lends itself well to the cause of specifying properties of properties, or meta-properties. Fig. 1 illustrates a property definition in RDFS.

Each of the meta-properties is defined in RDF as a property having a property and a string literal as its domain and range as shown in Figure 2. The use of relation elements in this way to describe relations would not be possible without reification or the ability to treat properties in the same way as concepts. This is not the case in current state-of-the-art KR formalisms such as DAML, OIL and OWL, which are based upon DL systems that do not support reification. Moreover, DL systems do not provide constructs to define properties in the same breadth of detail as classes.



```
<rdf:Property rdf:ID=" subEvent">
   <rdfs:comment> subEvent(Event1, Event2)means Event1 is a
                      subEvent of Event2
    </rdfs:comment>
     <rdfs:domain rdf:resource="#Event"/>
     <rdfs:range rdf:resource="#Event"/>
    <uofM_eece:composable>   yes
     </uofM_eece:composable>
    <uofM_eece:connected>    yes      </uofM_eece:connected>
     <uofM_eece:functional>   yes
     </uofM_eece:functional>
     <uofM_eece:homeomerous>  no
     </uofM_eece:homeomerous>
     <uofM_eece:intangible>   no
     </uofM_eece:intangible>
     <uofM_eece:intrinsic>    yes     </uofM_eece:intrinsic>
     <uofM_eece:near>         yes     </uofM_eece:near>
     <uofM_eece:separable>    no      </uofM_eece:separable>
     <uofM_eece:structural>   n/a
     </uofM_eece:structural>
     <uofM_eece:temporal>     no      </uofM_eece:temporal>
</rdf:Property>
```

Fig. 1. Definition of a property using a set of meta-properties in RDF/RDFS



```
<rdf:Property rdf:ID="composable">
   <rdfs:domain rdf:resource="#subEvent"/>
   <rdfs:range  rdf:resource="&xsd;StringLiteral"/>
</rdf:Property>
```

Fig. 2. Definition of a meta-property in RDF/RDFS

Constructs from OWL and DAML can capture inclusion relationships among relations, apart from the transitivity, asymmetry and reflexivity of relations but not reification. This is the reason that relation element semantics are not supported in such DL-based KR paradigms. An extension to the underlying DL system to treat relations in the same level of detail as concepts would enable the definition of relation elements using OWL. Note that relations correspond to object properties in OWL and DAML or simply properties in RDF and RDFS. These richer relation semantics would subsequently benefit inferencing mechanisms for Semantic Web applications.

In the next section, some of the work in relation element theory is summarized and the benefits of relation elements from the perspective of the Semantic Web are discussed.

3 Relation Elements for the Semantic Web

Relation elements were first proposed by Chaffin and Hermann [13]. Relation elements are used to describe relations and can also be used for the analysis and classification of relations. This is because relation elements explicitly capture the intrinsic nature of the relationship between the domain and range entities.

It is proposed that using relation elements to describe properties of relations or object properties on the Semantic Web will have quite a few benefits for automated inference. Expressing these properties of relations is beyond the capabilities of current state-of-the-art Semantic Web KR formalisms. In the following subsections, some of these benefits of using relation elements on the Semantic Web are enumerated and discussed.

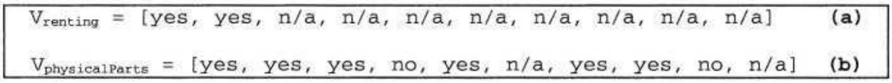
3.1 Plausible Inference for the Semantic Web Using Relation Elements

Huhns and Stephens [11] defined a set of relation elements on the basis of previous work by Chaffin and Hermann [13], Winston et al. [14] and Cohen and Loiselle [15]. These relation elements can be used to derive plausible inferences from previously asserted knowledge in a knowledge base.

Plausible inference mechanisms infer new knowledge that is not within the deductive closure or logical entailment of previously asserted knowledge. This involves the composition of existing relation instances to infer new relation instances between the participating entities. The relation elements that describe the relations involved are the basis of the composition of these relation instances.

Recall that every relation element takes on a value from {*yes, no, n/a*} for a relation that it describes. Therefore, a relation can be defined as a vector of relation element values. In Figure 3a for example, the relation *renting* is defined by a vector of ten relation element values as shown. Figure 3b shows the relation *physicalParts* defined using a different vector of relation element values.

When two relation instances are composed, a new relation instance is derived and consists of a new vector of relation element values. The relation element values of the new relation instance are derived by combining the corresponding relation element values of the composed relation instances. Huhns and Stephens [11] also proposed an algebra for combining the corresponding relation element values in their work.



$V_{renting}$ = [yes, yes, n/a, n/a, n/a, n/a, n/a, n/a, n/a, n/a] **(a)**

$V_{physicalParts}$ = [yes, yes, yes, no, yes, n/a, yes, yes, no, n/a] **(b)**

Fig. 3. The *renting* and *physicalParts* relations represented as vectors of relation element values.

Figure 4 shows an example of composing two relation instances on the basis of their relation element values. In Figure 4, composing the *renting(person, car)* instance with the *physicalParts(car, engine)* instance yields a set of relation element values that are identical to that of the *renting* relation. In this way, the inference that a person renting a car also rents the engine of the car can be drawn using relation elements. This inference is beyond the deductive closure of the knowledge previously asserted and is not entailed by logical deduction inference mechanisms. This example is summarized by Figure 5.

```
renting(Person, Car) O physicalParts(Car, Engine) =
```

$V_{renting}$ O $V_{physicalParts}$ =

```
 [yes,  yes,  n/a,  n/a, n/a,  n/a, n/a, n/a,  n/a, n/a]
 [yes,  yes,  yes,  no,  yes,  n/a, yes, yes,  no,  n/a]
-------------------------------------------------------------
 [yes,  yes,  n/a,  n/a, n/a,  n/a, n/a, n/a,  n/a, n/a]
```

Fig. 4. Composition of two relations using relation elements.

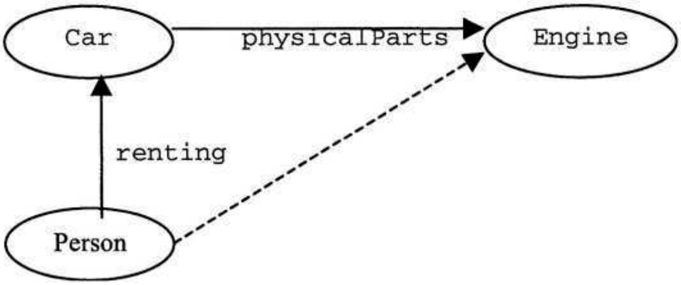


Fig. 5. Deriving a new relation instance using plausible inference.

Figure 5 shows the two composed relation instances as edges between nodes that correspond to the entities that they relate. The dashed arrow represents the relation instance between the previously unrelated entities that is inferred by plausible inference. The interested reader is referred to [12] for the details of an implementation of plausible inference for the Semantic Web that uses the set of relation elements proposed by [11].

3.2 A Top-Level Relation Hierarchy Using Relation Elements

A top-level hierarchy of relations would be analogous to high-level ontological representations such as Allen's temporal logic [16] and Sowa's upper ontology [17] that outline the hierarchy of the most basic and common concepts of all human knowledge.

Research initiatives such as Cyc [18] and the IEEE Suggested Upper Merged Ontology (SUMO) [19, 20] are Semantic Web compatible ontological representations that capture the taxonomy of these basic concepts within the dimensions of time and space. The concepts in a high-level ontology such as SUMO can be extended by more domain specific ontological representations and thus enables a top-down approach to the process of knowledge sharing and reuse across the Semantic Web. It is proposed that the development of a similar Semantic Web compatible taxonomy of high-level relations would further the cause of knowledge sharing and reuse.

Chaffin and Hermann [13] identified 31 different kinds of relations on the basis of the nature of the relationship between the domain and the range, grouping them into 5 families. These are: *contrasts, similars, class inclusion, case relations* and *part-whole.* They also proposed different sets of relation elements to describe relations from different families. The set of relation elements to describe relations in the *part-whole* family is for instance different from the set for the relations from the *contrasts* family. The set of relation elements particular to a family also forms a hierarchy. The top relation element in this hierarchy (the root) serves to distinguish the family from the other families. The relations within a family are distinguished by relation elements at deeper levels of the hierarchy. This could be a good starting point for the classification of relations into a taxonomic hierarchy. However, Hermann and Chaffin contend that the list of 31 relation types they describe is by no means exhaustive; nor is the list of sets of relation elements adequate to describe every relation.

It would be very difficult to derive an exhaustive and universally agreed upon list of relation element sets to describe every relation in every knowledge domain. Few relation elements would be relevant across all knowledge domains. For instance, the *homeomerous* relation element would not be relevant to describe relations in the domain of atomic physics. Even within knowledge domains, not all relation elements would be relevant across all contexts. Therefore, a more pragmatic task would be to restrict the scope of the relation elements to a knowledge domain of interest or further, to a context within a knowledge domain.

Kashyap and Borgida [21] have categorized relations in the medical domain into 5 classes on the basis of the nature of the relationship between the domain and the range. These are: *physically_related_to* as in the domain is physically related to the range, *spatially_related_to, conceptually_related_to, temporally_related_to* and *functionally_related_to.* These can be used as a set of relation elements to describe and classify relations within the medical domain.

To handle contexts, Cyc is divided into locally consistent contexts or microtheories, as pointed out by Reed and Lenat [22]. Each microtheory contains assertions and the set of assumptions (or the context) under which the assertions are valid. Therefore, some relation elements would have to be declared and used in conjunction with the contexts where they are of relevance.

Similarly, the set of relation elements proposed by [11] are by no means adequate to represent every known relation. However, the relations that they can describe can be considered to be a part of a microtheory. Within the scope of this microtheory, the relation elements can be classified into those that specify a meronymic or part-whole relationship between the domain and range and those that do not. This classification can be done on the basis of the values that can be assigned to an element. For example, in non-meronymic relations, the elements *functional, homeomerous* and *separable* are assigned the value *n/a.* In meronymic relations these relation elements are assigned the values *yes* or *no.*

Winston et al. [14] and Priss [23] have used the relation element theory to further subdivide meronymic relations on the basis of the nature of the relationship between the part and the whole. Storey [24] uses a slightly different approach to classify meronymic relations into 7 different categories viz. *component-object, member-collection, portion-mass, stuff-object, phase-activity, place-area* and *feature-event.*

Of the three relation elements describing meronymic relations, the *homeomerous* relation element is not required to describe meronymic relations belonging to the *phase-activity, place-area, member-collection, component-object* and *feature-event* categories. On this basis, the set of relation elements describing meronymic relations can be subdivided into *homogeneous* and *heterogeneous* categories. The *homeomerous* relation element has the value *yes* in the *homogeneous* category and the value *no* in the *heterogeneous* category.

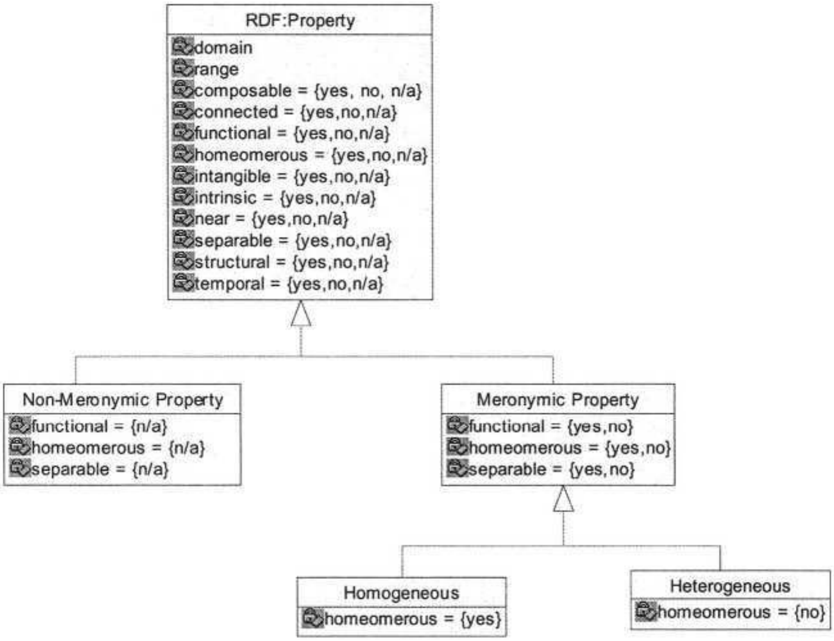


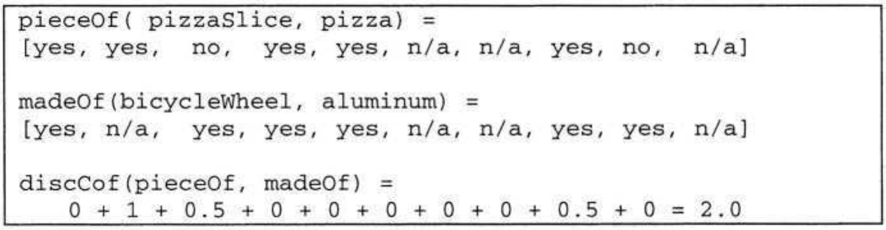
Fig. 6. Partition of the relation elements proposed by Huhns and Stephens

The relation elements in the *non-meronymic* category can be subdivided in a variety of ways. For instance, they can be subdivided into those that specify a temporal relationship between the domain and the range elements and those that do not. The relation element *temporal* would be irrelevant in describing relations that do not specify a temporal relationship between the domain and range elements and as such would have the value *n/a* in one of the two categories. Figure 6 depicts the partition of the relation elements proposed by [11]. The partitioning of the relation elements in the *non-meronymic* category can be done in several ways and is not taken up in detail here. The complications that would arise in the partitioning or classification of relation elements capable of describing every possible relation can be envisaged from this simple example. This is why the restriction of the scope of relation elements to contexts (or microtheories) or to knowledge domains is a much more pragmatic option.

3.3 Mapping Relations Between Ontologies Using Relation Elements

In a distributed environment such as the Semantic Web, several different ontological representations can exist for a knowledge domain. Problems arise during the integration of these different ontologies due to synonymy and polysemy, which hinder knowledge sharing and reuse. Specifically, a relation *partOf* relating an entity *Engine* to another entity *Car* can be defined synonymously in a different ontology (describing the same knowledge domain) as *physicalParts*. A machine without prior definitions cannot infer that the semantics of these two relations are identical. Klein [25] and Doan et al. [26] describe the inter-ontology mapping and integration problem in detail and propose several possible solutions.

It is proposed that relation elements can be used to infer the semantic similarity of two synonymous relations such as the ones above, thus aiding the ontology mapping and integration process. Consider the two relations *pieceOf* and *madeOf*. The *pieceOf* relation relates an entity *PizzaSlice* to another entity *Pizza*. The *madeOf* relation relates an entity *BicycleWheel* to another entity *Aluminium*. Both of these relations can be represented as a vector of relation element values as shown in Figure 7.



```
pieceOf( pizzaSlice, pizza) =
[yes, yes,  no,  yes, yes, n/a, n/a, yes, no,  n/a]

madeOf(bicycleWheel, aluminum) =
[yes, n/a,  yes, yes, yes, n/a, n/a, yes, yes, n/a]

discCof(pieceOf, madeOf) =
   0 + 1 + 0.5 + 0 + 0 + 0 + 0 + 0 + 0.5 + 0 = 2.0
```

Fig. 7. Computing the discriminator coefficient

Comparing the value of each relation element in turn, a score of 0 is assigned if the values are identical, +1 if one of the values is n/a and the other is either yes or no, and +0.5 if both the values are either yes or no but not identical. A discriminator coefficient on a scale between 0 and 10 can be computed by summing up the scores across the two sets of relation element values. A higher coefficient would indicate a greater degree of dissimilarity between the two relations. In the example in Figure 7, a coefficient of 2.0 is computed between the *pieceOf* and *madeOf* relations indicating a significant amount of semantic similarity between them.

This is a naive approach to computing similarity that uses the simple set of relation elements proposed by Huhns and Stephens [11]. A more rigorous methodology is based upon the taxonomy of relation elements proposed by Chaffin and Hermann [13]. To expound briefly, two relations could be compared at different levels of the relation element hierarchy starting from the top level and then progressing towards the bottom. Assigned scores would be weighted on the basis of the level of hierarchy at which the relation element values differ.

Algorithms similar to this can be developed to quantitatively specify the similarity or distinctness of pairs of relations on the basis of the values of their underlying relation elements. The same would apply for the translation of relations between global and contextual ontological representations [27] that differ in definition scope or as a part of "relative decontextualization" [28].

4 Conclusions

The investigation into the intrinsic nature of relations goes back to the times of Aristotle. Relation element theory attempts to categorize relations on the basis of the relationship between the domain and the range entities that the relation captures. While not adequate to describe every known relation, relation elements offer a powerful methodology to define the intrinsic semantics of relations facilitating the creation of relation taxonomies and disambiguation among relations. Upper level ontologies and knowledge representation formalisms for the Semantic Web focus more upon the nature of concepts in a knowledge domain and focus relatively little upon that of relations between these concepts. The use of relation elements to create relation taxonomies can help overcome this limitation.

In this paper, the limitations of state-of-the-art knowledge representation formalisms such as OWL and DAML for capturing intrinsic semantics of relations have been highlighted. Extending the underlying DL system to provide richer semantics including reification of properties to overcome this shortcoming is the subject of our current research. An initial methodology using RDF and RDFS constructs to specify properties of relations beyond the capabilities of current state-of-the-art Semantic Web KR formalisms such as OWL and DAML has been presented.

Some of the possible benefits of incorporating additional relation semantics into KR formalisms for the Semantic Web have been discussed at length. Specifically, an implementation of plausible inference for the Semantic Web, which leverages the additional semantics provided by relation elements, has been presented. A naive algorithm that uses relation element values to quantitatively determine the semantic similarity or dissimilarity of relations has also been introduced. This approach has been presented as a possible solution to the problems encountered in the merging and integration of different ontologies pertaining to the same knowledge domain or in the process of relative decontextualization. Finally, the importance of precise relation semantics by way of relation elements for the creation of a top-level relation hierarchy has also been discussed in some detail.

A caveat in the relation element approach is the dependence of the relation element values on the subjectivity of the ontological engineer who is responsible for making the assertions. This can be overcome by consensus standards and with deployment experience over time.

Possibilities for continued work include investigations into extending DL systems to support richer relation semantics and reification such that relations and associations among relations can be incorporated using constructs in languages such as OWL and DAML. This extension will enable the specification of relations with semantics similar to concepts. Another area for continued work is the specification of a high-level relation taxonomy for the Semantic Web similar to the class hierarchy of IEEE SUMO or Cyc. Lastly, a more rigorous method for the determination of the semantic similarity of relations based upon the relation element taxonomy proposed by [13] will be taken up in the future.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American. (May 2001).

[2] Lassila O., Swick R.: Resource Description Framework (RDF) Model and Syntax Specification: W3C Recommendation (February 1999). Available online: http://www.w3.org/TR/REC-rdf-syntax/ (current February 2003).

[3] Brickley, D., Guha, R. V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft (January 2003). Available online: http://www.w3.org/TR/rdf-schema/ (current February 2003).

[4] Pan, J. Z., Horrocks, I.: RDFS (FA) and RDF MT: Two Semantics for RDFS. In: Fensel, D., Sycara, K., Mylopoulos, J., (Eds.): The Semantic Web – ISWC 2003. Proceedings of the **2nd** International Semantic Web Conference. LNCS 2870, Springer Verlag, Sanibel Island, FL (October 2003). 30 – 46.

[5] DAML Joint Committee: DAML+OIL. (2001) Online Resource: http://www.daml.org/2001/03/daml+oil-index.html (current November 2002).

[6] Fensel, D. et al.: OIL in a Nutshell. In Dieng, R. et al. (Eds.): Knowledge Acquisition, Modeling and Management. Proceedings of the European Knowledge Acquisition Conference (EKAW-2000). LNAI, Springer Verlag. (October 2000).

[7] Smith, M. et al.: OWL Web Ontology Language Guide: W3C Proposed Recommendation 15 December 2003. Online Resource: http://www.w3.org/TR/2003/PR-owl-guide-20031215/ (December 2003).

[8] Bechhofer, S. et al.: OWL Web Ontology Reference: W3C Proposed Recommendation 15 December 2003. Online Resource: http://www.w3.org/TR/2003/PR-owl-ref-20031215/ (December 2003).

[9] Baader, F., et al.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press. (March 2003).

[10] Kothari, C. R., Russomanno, D. J.: Modeling Logic-Based Constraints in OWL. In: Proceedings of IASTED International Conference on Artificial Intelligence and Applications (AIA 2004). Innsbruck, Austria. (February 2004). 600 – 608.

[11] Huhns, M., Stephens, L.: Plausible Inferencing Using Extended Composition. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Detroit, MI. (August 1989). 1420–1425.

[12] Russomanno, D. J., Kothari, C. R.: An Implementation of Plausible Inference for the Semantic Web. In: Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE 2003). Las Vegas, NV. (June 2003). 246 – 251.

[13] Chaffin, R., Hermann, D.: Relation Element Theory: A New Account of the Representation and Processing of Semantic Relations. In: Gorfein, D., Hoffman, R. (Eds.): Memory and Learning: The Ebbinghaus Centennial Conference. Lawrence- Erlbaum. Hillsdale, NJ. (1987).

[14] Winston, M. E., Chaffin, R., Hermann, D.: A taxonomy of part-whole relations. Cognitive Science. 11(4): 417 – 444. (1987).

[15] Cohen, P., Loiselle, C.: Beyond ISA: Structures for Plausible Inference in Semantic Networks. In: Proceedings of the Seventh National Conference on Artificial Intelligence. St. Paul, MN. (August 1988). 415 – 420.

[16] Allen, J.: Towards a General Theory of Action and Time. Artificial Intelligence. 23 (1984). 123–154.

[17] Sowa, J.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole. Pacific Grove, CA. (2000).

[18] Lenat, D. B.: Cyc: A Large-Scale Investment in Knowledge Infrastructure. Communications of the ACM, Vol. 38, No. 1. (November 1995).

[19] Niles, I., Pease, A.: Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology. In: Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology. Seattle, WA. (August 6, 2001).

[20] Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: Welty, C., Smith, B. (Eds.): Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001). Ogunquit, ME. (October 2001).

[21] Kashyap, V., Borgida, A.: Representing the **UMLS**® Semantic Network using OWL. In: Fensel, D. et al. (Eds.): The Semantic Web – ISWC 2003. Proceedings of the **2nd** International Semantic Web Conference. Sanibel Island, FL. LNCS 2870, Springer Verlag. (October 2003). 1–16.

[22] Reed, S. L., Lenat, D. B.: Mapping Ontologies into Cyc. AAAI 2002 Conference Workshop on Ontologies for the Semantic Web. Edmonton, Canada. (July 2002). Available online at: http://www.cyc.com/doc/white_papers/mapping-ontologies-into-cyc_v31 .pdf

[23] Priss, U.: Classification of Meronymy by Methods of Relational Concept Analysis. In: Proceedings of the Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 96). Bloomington, IN. (1996). Available online at:
http://www.cs.indiana.edu/event/maics96/Proceedings/Priss/priss.html

[24] Storey, V. C.: Understanding Semantic Relationships. In: Very Large Databases Journal, Vol. 2, Issue 4. 455 – 488. (1993).

[25] Klein, M.: Combining and relating ontologies: An analysis of problems and solutions. In: Gomez-Perez, A. et al. (Eds.): Workshop on Ontologies and Information Sharing IJCAI 2001. Seattle, WA. (2001).

[26] Doan, A. et al.: Learning to Map between Ontologies on the Semantic Web. In: Proceedings of the 11th International World Wide Web Conference (WWW 2002). Hawaii, USA. (2002).

[27] Bouquet, P. et al.: C-OWL: Contextualizing Ontologies. In: Fensel, D. et al. (Eds.): The Semantic Web – ISWC 2003. Proceedings of the **2nd** International Semantic Web Conference. Sanibel Island, FL. LNCS 2870, Springer Verlag. (October 2003). 164–179.

[28] Guha, R. V.: Contexts: A formalization and some applications. Ph.D. Dissertation. Stanford University. (1991).

Poset Ontologies and Concept Lattices as Semantic Hierarchies

Cliff Joslyn

Computer and Computational Sciences
Los Alamos National Laboratory
joslyn@lanl.gov

Abstract. We describe some aspects of our research in relational knowledge discovery and combinatorial scientific computing [11], with special emphasis on the relation to the research portfolio of the conceptual structures community. We have recently been developing [10,12] a combinatorial approach to the management and analysis of large ontologies such as the Gene Ontology (GO) [6]. Our approach depends on casting the GO as a labeled partially ordered set (poset) [16], and then using scores based on pseudo-distance measures which we have developed to categorize lists of labels (in the case of the GO, genes and gene products) concerning their clustering and depth within the GO. We hold that such taxonomic semantic hierarchies serve as the core conceptual structures underlying all ontological databases, and through this work we have developed a number of what we believe to be both fundamental and novel ideas about treating such large posets as data objects, in particular the nature of *distance* in such structures, and the nature of *level* as an *interval-valued* property. After laying out this basic framework, we can then bring these ideas to a particular kind of poset, namely the concept lattice [5]. Considering a concept lattice as a poset, we are then prepared to develop techniques for anomaly detection in relational data by measuring the relative level of concepts vs. their cardinalities.

1 Introduction

Semantic hierarchies are ubiquitous, not just in formal semantic structures like conceptual graphs, ontologies, and concept lattices (CLs), but also in metamodeling environments, object-oriented typing architectures, and even natural and computational linguistics. We are concerned with the fundamental nature of semantic hierarchies, and report on the current state of our work here.

We open with some discussion of the POSet Ontology Categorizer (POSOC), which was the motivation for the beginning of this work. POSOC was in turn motivated by the need for biologists to use algorithmic tools to navigate the Gene Ontology (GO), the best example of the vast, novel conceptual structures which the genomic revolution has thrust into the world only very recently: very large, taxonomically organized, hierarchical data objects as specialized databases.

Our view is that semantic hierarchies naturally live within the theory of partially-ordered sets (posets), and POSOC was developed on that basis. After

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 287–302, 2004.

reviewing POSOC's foundations, including some elementary partially ordered set (poset) theory, we then move on to consider general semantic hierarchies, and thus arrive at our core points: explicating our new conceptualizations of level and distance in posets as a vector-valued quantity of the height and width of the neighborhood (defined in a particular way) of a collection of poset nodes.

We conclude with some speculations about the use of such concepts in lattices, particularly CLs, conceived of as proto- or putative ontologies generated in the context of available relational knowledge, in our case, protein-ligand binding.

2 The POSet Ontology Categorizer (POSOC)

The computational biology revolution has produced a proliferation of large databases of genomic information. A premier example is the Gene Ontology (GO)[1] [6], a large (> 16,000 node), standardized knowledge structure consisting of three branches: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC). Each branch is organized as a taxonomy of nodes which represent different categories of genomic characteristics. Once a gene is sufficiently characterized, it can be attached to the appropriate node, as shown in Fig. 1 [6].

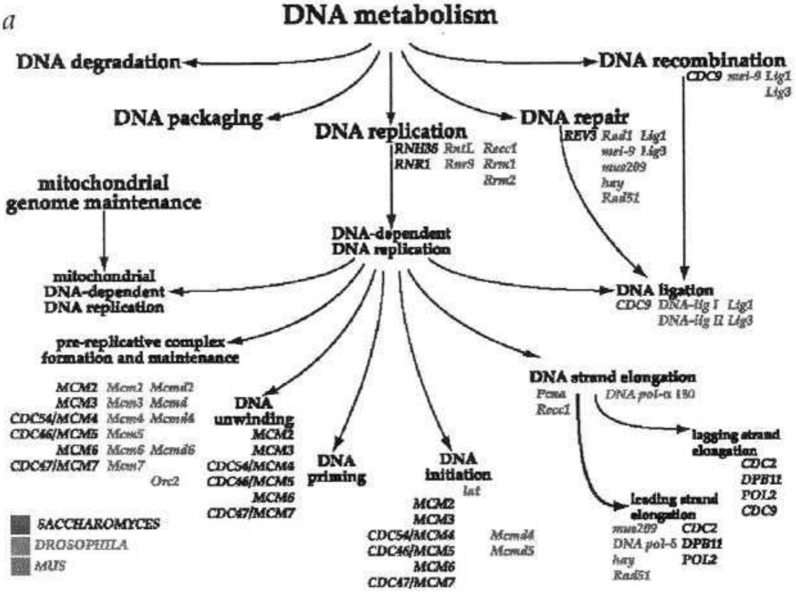


Fig. 1. A portion of the BP branch of the GO [6]. GO nodes in the hierarchy have genes from three species annotated below them.

[1] http://www.geneontology.org

We have been working on the **categorization** task in the GO, where following a gene expression experiment involving high throughput microarrays or Affymetrix gene chips, a biomedical researcher is confronted with a list of a few hundred to a thousand genes, from which she will need to extract useful information about the various types of biological processes that were affected in the experiment. The researcher then wants to take the names of these genes which have been annotated to the GO and gain an understanding of their overall function by examining their distribution through the GO: are they localized, grouped in distinct areas, or spread uniformly? Manual approaches and existing software are inadequate to answer this question over hundreds of proteins and more than 16,000 GO nodes, and thus an algorithmic approach is necessary.

At its core, the GO is a hierarchy of semantic categories. So to approach this problem, we have needed to address a number of fundamental questions about the nature of such hierarchies, modeled as partially ordered sets (posets), to provide algorithmically determined numerical scoring of the nodes in the GO with respect to the genes of interest. We produce a ranked list of appropriate summarizing nodes within the GO, which act as functional hypotheses about the characteristics of the genes expressed.

POSOC has been developed over the past year [9,10,11,12] by researchers at the Los Alamos National Laboratory (LANL) and Procter & Gamble Corp. (P&G), and is currently in use by staff scientists at P&G[2]. In addition, extensions of POSOC to handle textually-based queries have been used recently in a submission by LANL for the BioCreative challenge[3] for automated annotation [18].

2.1 Posets

We first introduce some elementary ideas from the theory of **finite partially ordered sets** (posets). This is mostly standard and elementary [16], but in some cases novel (to our knowledge), at least in terms of notation and perspective.

A finite poset is a structure $\mathcal{P} = \langle P, \leq \rangle$ where P is a finite set and $\leq \subseteq P^2$ is a reflexive, anti-symmetric, transitive binary relation on P. Posets are the most general combinatorial structures admitting to description in terms of *levels,* in our case, levels of semantic generality. While more specific than directed graphs or networks (every poset is a digraph with no cycles), they are more general than trees or lattices (every tree and lattice is a poset), in that collections of nodes can have multiple parents.

The GO is notably a directed acyclic graph (DAG), as is evident in Fig. 1, and every DAG determines both a unique poset and a unique Hasse diagram, in which all transitive links have been removed[4]. In a poset, two nodes $p, q \in P$

[2] As previously reported [9,10,12], POSOC was originally targeted specifically at the GO, and was thus called the Gene Ontology Categorizer (GOC). GOC has now been generalized to deal with any poset ontology, and is thus now called POSOC here.

[3] http://www.mitre.org/public/biocreative

[4] I.e., if both $a \leq b$ and $b \leq c$ are included, then if $a \leq c$ is present, it is removed.

are **comparable,** denoted $p \sim q$, if either $p \le q$ or $p \le q$; a **chain** $C \subseteq P$ is a collection of comparable nodes; and the **height** $\mathcal{H}(\mathcal{P})$ is the size of the largest chain. Similarly, two nodes $p, q \in P$ are **non-comparable** if $p \not\sim q$, an **anti-chain** is a collection of non-comparable nodes, and the **width** $\mathcal{W}(\mathcal{P})$ is the size of the largest anti-chain. For any node $p \in P$, its **ideal** is $\downarrow p := \{q \in P : q \le p\}$, its **filter** is $\uparrow p := \{q \in P : q \ge p\}$, and its **hourglass** is $\Xi(p) := \uparrow p \cup \downarrow p$. Define these concepts over a collection of nodes $Q \subseteq P$ similarly:

$$\downarrow Q := \bigcup_{p \in Q} \downarrow p, \quad \uparrow Q := \bigcup_{p \in Q} \uparrow p, \quad \Xi(Q) := \bigcup_{p \in Q} \Xi(p).$$

For any subset $Q \subseteq P$, a node $p \in Q$ is **maximal** in Q if $\nexists q \in Q, q > p$. Let Max(Q) be the set of all maximal nodes in Q, noting that Max(Q) must be non-empty if Q is non-empty. Define the set of all minimal nodes Min(Q) dually. For any two nodes $p, q \in P$ the set $\uparrow p \cap \uparrow q$ is their "joint filter" in some sense, and $p \vee q := \text{Min}(\uparrow p \cap \uparrow q)$ are their **joins.** For a collection of nodes $Q \subseteq P$, let

$$\bigvee Q := \text{Min}\left(\bigcap_{p \in Q} \uparrow p \right).$$

Lower bounds and meets $\wedge$ are defined dually. Note that posets are distinguished from lattices in that $p \vee q \subseteq P$ is not a single node, and is not guaranteed to exist, but is rather an arbitrary, possibly empty, subset of nodes.

If there exists a node $1 \in P$ such that $\text{Max}(P) = \bigvee P = \{1\}$, then we say that $\mathcal{P}$ is **upper-bounded,** and dually for $0 \in P$. If either there is no unique upper or lower bound $0, 1 \in P$, then we can create them easily by constructing the **closure** of $\mathcal{P}$ as $\bar{\mathcal{P}} := \langle P \cup \{0,1\}, \bar{\le} \rangle$, where $\forall p, q \in P, p \bar{\le} q \leftrightarrow p \le q$, and $\forall p \in P, 0 \bar{\le} p \bar{\le} 1$. Most of our results below require either an upper-, lower-, or totally bounded poset. We will presume that when $\mathcal{P}$ is not naturally so bounded, its closure $\bar{\mathcal{P}}$ is available in this way.

For two comparable nodes $p \le q$, all the nodes "between" them is the **interval** $[p, q] = \{t : p \le t \le q\} = \uparrow p \cap \downarrow q$. For comparable subsets $P_1, P_2 \subseteq P$ with $P_1 \le P_2$ (so that $\forall p \in P_1, q \in P_2, p \le q$), their interval $[P_1, P_2]$ is

$$[P_1, P_2] := \bigcup_{\langle p_1, p_2 \rangle \in P_1 \times P_2} [p_1, p_2].$$

For two comparable nodes $p \le q$, the interval $[p, q]$ is equivalent to the set of all chains between p and p, denoted $C(p, q)$. The **vector of chain lengths** $\boldsymbol{h}(p, q) := \langle |C(p, q)| \rangle$ is the collection of the lengths of all these chains, and finally the minimal and maximum chain lengths between p and q respectively are $h_*(p, q) := \min_{C \in C(p,q)} |C|$ and $h^*(p, q) := \max_{C \in C(p,q)} |C|$[5].

The Hasse diagram of an example of a poset on a set of nodes $P = \{1, A, B, \ldots, K\}$ is shown in Fig. 2. Note the inherently two-dimensional structure displayed by division into levels: while nodes can be re-drawn left to right (width)

[5] Here we assume the Hasse diagram, otherwise $p \le q \to h_*(p, q) = 1$.

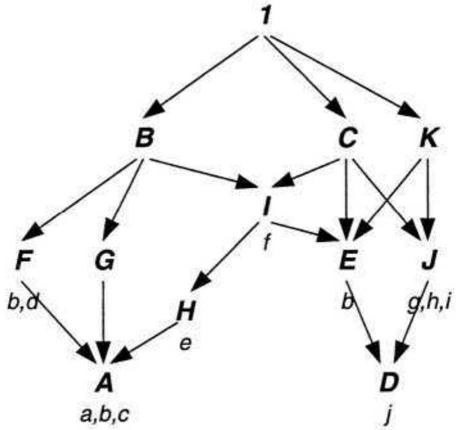


Fig. 2. An example of a labeled poset.

as convenient, vertically it's crucial that higher nodes be placed above lower ones (height).

2.2 The GO as a Labeled Poset

The GO has measurable poset properties, as shown in Tab. 1 and Fig. 3 (GO for September, 2003). The height parameter shows that the GO is properly seen as a structure divided into levels, 15 for BP and 13 for MF and CC. It branches out quickly and broadly, with twice as many nodes (10.6K) being "terminal" leaves compared to interior nodes (only 5.4K). Calculating the width of a poset is still daunting algorithmically, so the width shown here is only a lower-bound estimate. Thus the structure is at least three orders of magnitude wider than it is high. Fig. 3 shows the distribution (on a log scale) of the number of parents and children per node. Note that a few nodes have hundreds of children, and a substantial quantity have at least two parents, some as many as four or five.

We can then define a structure $\mathcal{O} := \langle \mathcal{P}, X, F \rangle$ as a **POSet Ontology (POSO),** where X is a finite, non-empty set of **labels,** and $F\colon X \mapsto 2^{P}$ is a function mapping each label $x \in X$ to a collection of nodes $F(x) \subseteq P$. In Fig. 2,

Table 1. Poset statistics of the GO.

	Nodes	Leaves	Interior	Edges	$\mathcal{H}$	$\mathcal{W}$
MF	7.0K	5.6K	1.3K	8.1K	13	$\geq$ 3.5K
BP	7.7K	4.1K	3.6K	11.8K	15	$\geq$ 2.9K
CC	1.3K	0.9K	0.4K	1.7K	13	$\geq$ 0.4K
GO	16.0K	10.6K	5.4K	21.5K	16	$\geq$ 5.9K

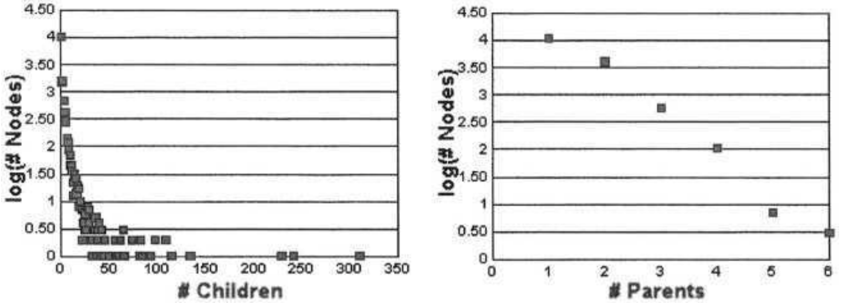


Fig. 3. Distribution of number of children (left) and parents (right) per node.

we have $X = \{a, b, \ldots, j\}$, and e.g. $F(b) = \{A, E, F\}$. In the case of the GO, then P is the collection of GO nodes, $\leq$ is the ordering relations present in the GO, and X is the set of genes annotated to the GO, as illustrated in Fig. 1.

2.3 POSOC Methodology

We can now pose the categorization problem in the context of the example in Fig. 2: given a particular set of genes of interest cast as a query, say $Y = \{c, e, i\} \subseteq X$, what node(s) in P best summarize that set? One answer is C, since it "covers" all three genes, and does so in the most specific way. The node 1 also covers the genes, but would not be favored since it's a more general category. But it can also be argued that H is a good answer, since, while it only covers c and e, it does so more specifically than C does. We will see that this interplay between "coverage" and "specificity" will be central to the methodology developed.

To proceed, we need the concept of a **pseudo-distance** as a function $\delta \colon P^2 \mapsto \mathrm{I\!R}$ where $\forall p \leq q \in P, h_*(p,q) \leq \delta(p,q) \leq h^*(p,q)$; and a **normalized distance** as $\bar{\delta} := \delta / \mathcal{H}(\mathcal{P})$. Current pseudo-distances implemented in POSOC include: the **minimum path length** $\delta_m := h_*$; the **maximum path length** $\delta_x := h^*$; the **average of extreme path lengths** $\delta_{ax}(p,q) := \frac{h_*(p,q)+h^*(p,q)}{2}$; and the **average of all path lengths** $\delta_{ap}(p,p') := \frac{\sum_{h \in \boldsymbol{h}(p,q)} h}{|\boldsymbol{h}|}$. Other candidate pseudo-distances are in exploration.

Given a pseudo-distance and a set of nodes of interest $Y \subseteq X$, we then want to develop a **scoring function** $S_Y(p)$ which returns the weighted rank of a node $p \in P$ based on requested nodes Y. We actually use two kinds of scores, an **unnormalized score** $S_Y \colon P \mapsto \mathrm{I\!R}^+$ which returns an "absolute" number, and a **normalized score** $\hat{S}_Y \colon P \mapsto [0,1]$ which returns a "relative" number. We allow the user to choose the relative value placed on coverage vs. specificity by introducing a parameter $s \in \{\ldots -1, 0, 1, 2, 3, \ldots\}$, where low s emphasizes coverages, and high s emphasizes specificity. The scoring function can use either the unnormalized distance δ, or the normalized $\bar{\delta}$. Letting $r = 2^s$, we have the four scoring functions shown in Tab. 2.

Table 2. Scoring functions.

Distance	**Score**	
	Unnormalized	Normalized
Unnormalized	$S_Y(p) := \sum_{x \in Y} \sum_{p' \in F(x) : p' \leq p} \left(\delta^r(p', p) + 1\right)^{-1}$	$\hat{S}_Y(p) := \frac{S_Y(p)}{\sum_{x \in Y} \lvert F(x) \rvert}$
Normalized	$\bar{S}_Y(p) := \sum_{x \in Y} \sum_{p' \in F(x) : p' \leq p} \left(1 - \bar{\delta}(p', p)\right)^r$	$\hat{\bar{S}}_Y(p) := \frac{\bar{S}_Y(p)}{\sum_{x \in Y} \lvert F(x) \rvert}$

Table 3. POSOC output for example in Fig. 2 for query $Y = \{c, e, i\}$.

	$s = -1$		$s = 1$		$s = 3$	
Rank	$\hat{\bar{S}}_Y(p)$	p	$\hat{\bar{S}}_Y(p)$	p	$\hat{\bar{S}}_Y(p)$	p
1	0.7672	**C**	0.5467	**H**	0.3893	**H**
2	0.6798	**1***	0.3867	**C***	0.3333	A;**J**
3	0.6315	H	0.3333	A;I;J		
4	0.5563	I			0.0617	**C***
5	0.5164	B			0.0615	I
6	0.3333	A;J	0.2400	**B***	0.0559	F;G;K
7			0.2267	**1***		
8	0.2981	F;G;K	0.2133	F;G;K		
9					0.0112	B
10					0.0060	**1**

We then find non-comparable nodes within the ranked-list to serve as "cluster heads". The resulting clusters are at different depths in $\mathcal{P}$: while "headed" by non-comparable nodes, their contents (the collection of their descendants in $\mathcal{P}$) can overlap. Cluster heads which are non-comparable to all other cluster heads of lower rank are called "primary", and those above some previously identified cluster head "secondary".

Output for the example in Fig. 2 is shown in Tab. 3, for query $Y = \{c, e, i\}$, specificity values $s = -1, 1$, and 3, doubly-normalized score $\hat{\bar{S}}$, and pseudo-distance δ_m. Cluster heads are shown in bold, and secondaries are labeled with $*$. Inspection reveals desirable results: for low specificity, C is the preferred primary cluster, with 1 a secondary; for high specificity, H and J are preferred (J specifically covers i), with C as the next-ranked secondary.

POSOC was validated by a highly experienced molecular immunologist who had no prior knowledge of the POSOC to assess its utility and accuracy [12]. It was also validated formally by comparing POSOCs annotations to a collection of independent annotations of collections of GO nodes (corresponding to our lists of target genes) available through the InterPro project[6], which catalogs assignments of protein families, domains, and functional sites to GO IDs [12].

[6] `http://www.ebi.ac.uk/interpro`

As noted, we are in the process of generalizing POSOC's implementation to target any POSO, not just the GO. Current targets include the Enzyme Commission (EC) database[7] and the MEdical Subject Headings (MESH) ontology[8].

3 Requirements for Working with Semantic Hierarchies

While modern bio-ontologies take many forms, an adequate overall description is of a taxonomically organized data object over which automated inference and reasoning (for example using description logics [2]) is performed. Leading research in ontologies tends to focus on logical properties, inference, and search. Our view is that what has made existing bio-ontologies such as the GO so successful are their attributes as hierarchical, taxonomic, categorizations of biological objects, coming closer to being specially structured databases.

Moreover, these attributes are fundamental to other aspects: it is clear that large taxonomically-organized database can be very useful without an inference engine, but the converse is not so evident. Indeed, semantic hierarchies are truly ubiquitous. Even a casual observation reveals them at the foundations of knowledge architectures such as conceptual graphs [17], as object-oriented data types [14], in CLs and related work [5], and even in verb type hierarchies from cognitive linguistics [4]. And yet there seems to be little attention paid to the need for algorithmic approaches to their representation, analysis, navigation, manipulation, and measurement, or even their generic properties as formal structures.

While there are no doubt many reasons for this, these likely include the relatively later development of poset theory as compared to lattices and networks (the first serious textbook appeared in 2003 [16]), and especially the novel appearance of these large, taxonomically organized knowledge objects which now *require* this kind of computer-scientific approach.

So we are motivated to continue in a number of directions:

- First, we have found our pseudo-distances δ lacking, as they are only available between comparable nodes. We are thus seeking to generalize this idea to a more inclusive measures of distance, size, level, etc.
- There are many more tasks which need to be addressed within the overall poset ontology world than the categorization task. Examples include:

  Matching: How do we match two parts of a poset ontology? This arises, for example, in both the BioCreative task and the InterPro validation of POSOC, where POSOC has provided certain answers, and we wish to compare those to some "correct" answer provided by someone else. This can be formalized as follows: assume a poset $\mathcal{P} = \langle P, \leq \rangle$, with $P_1, P_2 \subseteq P$, inducing the sub-posets $\mathcal{P}_1 = \langle P_1, \leq|_{P_1} \rangle$ and $\mathcal{P}_2 = \langle P_2, \leq|_{P_2} \rangle$. How can we then measure the similarity of $\mathcal{P}_1$ and $\mathcal{P}_2$?

[7] http://www.biochem.ucl.ac.uk/bsm/enzymes

[8] http://www.nlm.nih.gov/mesh/meshhome.html

Comparison: Assume now that we have two different orderings available on the same underlying set, for example ontologies constructing by different teams of researchers. How can we compare their similarity? This can also be formalized as assuming $\mathcal{P}_1 := \langle P, \leq_1 \rangle$ and $\mathcal{P}_2 := \langle P, \leq_2 \rangle$, then how can we measure the similarity of $\mathcal{P}_1, \mathcal{P}_2$?

Merger: Finally we have the most general formulation of the problem, assuming two complete different ontologies $\mathcal{P}_1 := \langle P_1, \leq_1 \rangle$ and $\mathcal{P}_2 := \langle P_2, \leq_2 \rangle$. How can we hope to measure their similarity, and ultimately find ways to merge them together into some new poset $\mathcal{P}$ on $P_1 \cup P_2$?

The general situation is illustrated in Fig. 4, where the EC and the GO are shown as posets on different underlying sets *P,* but with the same set of labels *X.* This common labeling can also be used as a source of comparison information, showing, for example, similarity between nodes *A, G, F* of GO and *E, J* of EC in virtue of the annotation of genes b, g, h, i, some of which are analogous, and some of which (e.g. i) are not.

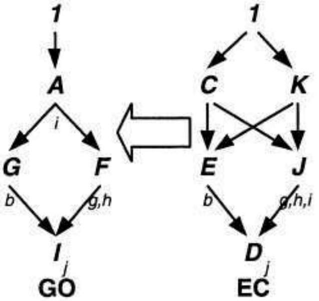


Fig. 4. Cartoon of the general ontology matching problem between the EC and GO.

- We are also interested in considering CLs as semantic hierarchies (see Sec. 5), and using formal measures of level and distance in them to induce hypotheses about both extractable knowledge and potential anomalies in data sets.
- Indeed, this general class of problems arises in a number of more specialized lattices and posets, for example posets of system reconstruction hypotheses in multi-dimensional statistical analysis [8,13,15] and classes of random sets in generalized information theory [7].

4 Measures in Semantic Hierarchies

In all these instances, what is required are much better conceptualizations of measures in posets. Our thoughts extend to two important concepts: a general, interval-valued concept of vertical **level** or **rank** within a poset; and a general, vector-valued concept of overall **distance** between two arbitrary nodes.

The scope for this paper allows only a partial formal development. Here we introduce a few suggestive definitions and results, and refer the reader to future work for a detailed development, including more proofs of the basic results.

4.1 Interval-Valued Poset Rank

Rank as a measure of the vertical "level" of a node is an important combinatorial concept [1,3], but usually used only in more constrained combinatorial structures such as lattices or so-called Jordan-Dedekind, or JD, posets[9]. We have found [16] rank to be defined in posets in a lower-bounded way:

$$r_*(p) := \begin{cases} 0, \ p \in \mathrm{Min}(P) \\ n, p \in \mathrm{Min}\,(P - \{q : r_*(q) < n\}) \end{cases}$$

We use r_* suggestively, as its dual function is readily available:

$$r^*(p) := \begin{cases} 0, \ p \in \mathrm{Max}(P) \\ n, p \in \mathrm{Max}\,(P - \{q : r^*(q) < n\}) \end{cases}$$

And so an interval rank function can be easily identified as $R \colon P \mapsto \mathcal{D}(\mathcal{H}(\mathcal{P}))$, with $R(p) := [r_*(p), \mathcal{H}(\mathcal{P}) - r^*(p)]$[10], where $\mathcal{D}(n) := \{[x,y] : x,y \in \mathcal{I}^+, 0 \leq x \leq y \leq n\}$ is the set of all integer intervals for $n \in \mathcal{I}^+$. An example of a bounded poset equipped with its interval rank function is shown in Fig. 5, so that $R(D) = [1,2], R(K) = [1,4]$.

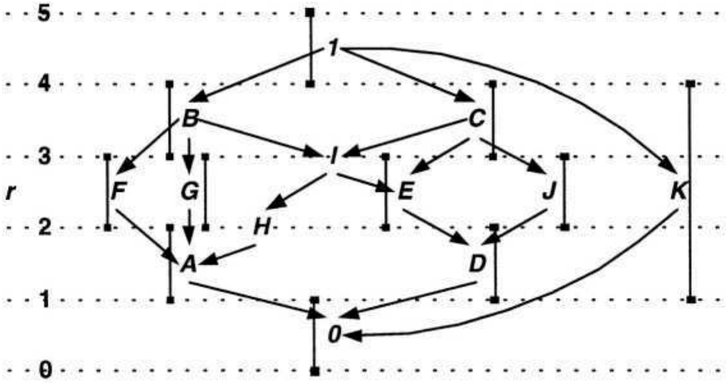


Fig. 5. A bounded poset equipped with its interval rank.

[9] Those where all chains between comparable nodes have the same length.

[10] Note that if we instead use $R(p) := [\mathcal{H}(\mathcal{P}) - r_*(p), r^*(p)]$, the "top" and "bottom" as in Fig. 5 are simply reversed.

To conceptualize the interval rank, first, $\mathcal{P}$ can possess at most $\mathcal{H}(\mathcal{P})$ "levels". So for any node $p \in P$, it's level has to be no less than as "high up" from the bottom as it is, but no more than how "far down" from the top it is.

Theorem 1. $R(p) = [h_*(0,p), \mathcal{H}(\mathcal{P}) - h_*(p,1)]$

Proof. Since $R(p) = [r_*(p), \mathcal{H}(\mathcal{P}) - r^*(p)]$, it is sufficient to show that $r_*(p) = h_*(0,p)$, and $r^*(p) = h_*(p,1)$. For $p = 0$, clearly $r_*(0) = 0 = h_*(0,0)$. So if $p \in \text{Min}(P - \text{Min}(P))$, then $r_*(p) = 1$, and also $h_*(0,p) = 1$. Indeed, at each step $\text{Min}(P - \{q : r_*(q) < n\}) = \{q : h_*(0,q) = n\}$, and thus r_* partitions P into bands of lower rank n with $h_* = n$. Thus $\forall p \in P, r_*(p) = h_*(0,p)$. $r^*(p) = h_*(p,1)$ follows by a dual argument.

R induces an order mapping $R\colon \mathcal{P} \mapsto \langle \mathcal{D}, \preceq \rangle$ from $\mathcal{P}$ to the set of integer intervals $\mathcal{D}$, where $\preceq$ is an interval order on $\mathcal{D}$. Whether R is order preserving depends on the interval order used. For two integer intervals $I = [I_*, I^*], J = [J_*, J^*] \in \mathcal{D}$, consider the following weak and strong interval orders:

$$I \preceq_w J := I_* \leq J_* \text{ and } I^* \leq J^*, \qquad I \preceq_s J := I^* \leq J_*.$$

Theorem 2. *R is order preserving from $\mathcal{P}$ to $\langle \mathcal{D}, \preceq_w \rangle$, but not to $\langle \mathcal{D}, \preceq_s \rangle$.*

Proof. Let $p \leq q$. Then $h_*(0,p) \leq h_*(0,q)$, and $h_*(q,1) \leq h_*(p,1)$. Thus we have $R(p) = [h_*(0,p), \mathcal{H}(\mathcal{P}) - h_*(p,1)] \preceq_w R(q) = [h_*(0,q), \mathcal{H}(\mathcal{P}) - h_*(q,1)]$ directly, but it might be that $\mathcal{H}(\mathcal{P}) - h_*(p,1) \leq h_*(0,q)$ or not, and so it could be that $R(p)$ and $R(q)$ are non-comparable in $\preceq_s$.

We also have the following unproved conjecture about how scalar-valued rank arises as a special case of our interval-valued rank.

Conjecture 1. Assume a fully bounded poset $\mathcal{P}$, and a node $p \in P$ with $r_*(p) = r^*(p)$ so that $R(p) = [r, r]$ for some specific $r \in \mathcal{I}$. Then $\forall C \in \mathcal{C}(0,1), p \in C$ iff C is maximal in the sense of $|C|$. Moreover, $\forall p \in P, R(p) = [r, r]$ iff $\mathcal{P}$ is JD.

For example, in Fig. 5, we have *R(H)* = [2, 2] = 2, and *H* is only on a maximal chain $0 \leq A \leq H \leq I \leq B \leq 1$.

4.2 Vector-Valued Poset Distance

In conjunction with our new sense of "vertical distance" in posets, we also wish to have a general sense of distance which captures the horizontal component as well. Towards that end, for some collection of nodes $Q \in P$, including both comparable and non-comparable pairs, we need to characterize the nodes "between" them in some sense. We characterize this as the **neighborhood** of *Q*, and our sense of distance is directly related to some measure of the "size" of this region of *P*. This should be a vector quantity consisting of a horizontal and vertical component, since these concepts are so distinct in posets.

We have some preliminary ideas in this direction, which we report here, although we regret that we haven't yet explored the implications of our definitions deeply yet, nor the relationship to interval-valued rank described above.

Definition 1 (Neighborhoods). *Assume a poset $\mathcal{P}$ and a collection of nodes $Q \in P$. If $\bigvee Q$ exists, then define the* **upper neighborhood** *of Q as the intersection of its filter and the ideal of its lubs:*

$$N^*(Q) := \uparrow Q \cap \downarrow \left(\bigvee Q\right).$$

If $\bigwedge Q$ exists, then define the **lower neighborhood** *of Q dually:*

$$N_*(Q) := \downarrow Q \cap \uparrow \left(\bigwedge Q\right).$$

When both $\bigvee Q$ and $\bigwedge Q$ exist, then define the **neighborhood** *as the intersection of the hourglass and the interval between the lubs and glbs of Q:*

$$N(Q) := \Xi(Q) \cap \left[\bigwedge Q, \bigvee Q\right].$$

In all cases, if $|Q| = 2$ so that $Q = \{p, q\}$, then define for each appropriate form e.g. $N(p,q) := N(Q)$.

Note that $N(Q)$ exists because necessarily $\bigwedge Q \leq \bigvee Q$. A simplified cartoon of the appearance of $N^*(p,q)$ is shown in Fig. 6.

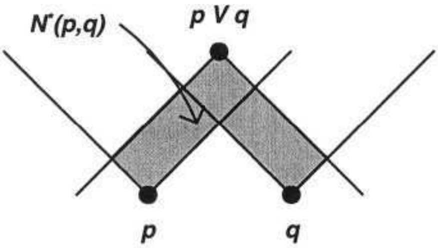


Fig. 6. Cartoon of the upper neighborhood $N^*(p,q)$ (shaded region).

The idea is to say that the nodes in the neighborhood of Q should be "entrained" by both the filter $\uparrow Q$ and ideal $\downarrow Q$ (that is, by the hourglass $\Xi(Q)$), but then also should not be "higher" than the joins $\bigvee Q$, nor "lower" than the meets $\bigwedge Q$; indeed, they should be only those parts of the hourglass between $\bigwedge Q$ and $\bigvee Q$. Thus we have:

Conjecture 2. $N(Q)$ is the set of all chains between $\bigwedge Q$ and $\bigvee Q$ which go through some node of Q.

Chains have no horizontal width, so an easy special case is recovered.

Theorem 3. *If $C = \{p_1, p_2, \ldots, p_n\} \subseteq P$ is a chain with $p_1 \leq p_2 \leq \ldots \leq p_n$, then $N(C) = [p_1, p_n]$.*

Proof. Let $C = \{p_1, p_2, \ldots, p_n\}$ be a chain with $p_1 \leq p_2 \leq \ldots \leq p_n$. Then $\uparrow p_1 \supseteq \uparrow p_i$ for all $2 \leq i \leq n$, and $\downarrow p_n \supseteq \downarrow p_i$ for all $1 \leq i \leq n-1$. Also, $\bigwedge C = p_1, \bigvee C = p_n$ both exist, so that $[\bigwedge C, \bigvee C] = [p_1, p_n]$. Thus we have:

$$\begin{aligned} N(C) &= \Xi(C) \cap \left[\bigwedge C, \bigvee C\right] \\ &= \left(\left(\bigcup_{i=1}^{n} \uparrow p_i\right) \cup \left(\bigcup_{i=1}^{n} \downarrow p_i\right)\right) \cap [p_1, p_n] \\ &= (\uparrow p_1 \cup \downarrow p_n) \cap (\uparrow p_1 \cup \downarrow p_n) = \uparrow p_1 \cup \downarrow p_n = [p_1, p_n]. \end{aligned}$$

Note the trivial corollary that $p \leq q \rightarrow N(p,q) = [p,q]$.

We now define a vector-valued distance in terms of these neighborhood.

Definition 2 (Size and Distance). *Assume a bounded poset $\mathcal{P}$. Then let the vector-valued size of a collection of nodes $Q \subseteq P$ be*

$$\mathbf{D}(Q) := \langle \mathcal{H}(N(Q)), \mathcal{W}(N(Q)) \rangle,$$

and the vector-valued distance between two nodes $p, q \in P$ be $\mathbf{D}(p,q) := \mathbf{D}(\{p,q\})$.

For examples, consider that in Fig. 2, we have

$$\begin{aligned} &N(B,J) = \Xi(B,J) \cap [B \wedge J, B \vee J] = (P - \{E\}) \cap [D,1] = [D,1], \\ &\mathbf{D}(B,J) = \langle \mathcal{H}([D,1]), \mathcal{W}([D,1]) \rangle = \langle 5,3 \rangle, \end{aligned}$$

and in Fig. 5, we have

$$\begin{aligned} N(J,K) &= \Xi(J,K) \cap [J \wedge K, J \vee K] = (\{0,D,J,C,1\} \cup \{0,K,1\}) \cap [0,1] \\ &= \{0,D,J,C,1,K\} \cap P = \{0,D,J,C,1,K\}, \\ \mathbf{D}(J,K) &= \langle \mathcal{H}(\{0,D,J,C,1,K\}), \mathcal{W}(\{0,D,J,C,1,K\}) \rangle = \langle 5,2 \rangle. \end{aligned}$$

We recover a pseudo-distance easily for the case of comparable nodes.

Theorem 4. *If $p \leq q$, then* $\mathbf{D}(p,q) = \langle \delta_x(p,q), 1 \rangle$.

Proof. If $p \leq q$, then we know from Thm. 3 that $N(p,q) = [p,q]$, and thus $\mathbf{D}(p,q) = \langle \mathcal{H}([p,q]), \mathcal{W}([p,q]) \rangle = \langle h^*(p,q), 0 \rangle = \langle \delta_x(p,q), 1 \rangle$.

In the future, we may recover other pseudo-distances δ if we first restrict our sense of height $\mathcal{H}(\mathcal{P})$ to bounded posets, but then relax it to be the interval $\mathcal{H}(\mathcal{P}) = [h_*(0,1), h^*(0,1)]$ instead of the scalar $\mathcal{H}(\mathcal{P}) = h^*(0,1)$.

5 Distance Measures in Concept Lattices

We recognize formal concept analysis (FCA) as both a foundational tool for the representation of relational information [5], and a way to extract semantic hierarchies from relational data. The trivial observation that every lattice is a poset opens the way to the consideration of the application of our ideas about levels and distances to nodes in CLs.

Space precludes a detailed exposition, instead we refer to the standard references [5]. Instead, we will simply assert the availability of a **context** as a binary relation $R \subseteq X \times Y$ which generates a poset $\mathcal{L} = \langle P, \leq \rangle$, where: $\mathcal{L}$ is actually a lattice, in particular the concept lattice of *R;* $P \subseteq 2^X \times 2^Y$ is a set of **concepts** generated by *R;* and $\leq$ is the subset ordering such that $p \leq q := p = \langle A_1, B_1 \rangle, q = \langle A_2, B_2 \rangle$, with $A_1 \subseteq A_2$ and $B_1 \supseteq B_2$, where $A_1, A_2 \subseteq X, B_1, B_2 \subseteq Y$.

Lattices as Special Posets: When a poset $\mathcal{P}$ is a lattice (recalling that *P* here is finite), we always have that $\forall Q \subseteq P, \bigvee Q$ and $\bigwedge Q$ exist, and furthermore as unique members of *P*. Thus the formulations of Def. (1) and (2) become largely simplified, for example:

$$N(p,q) = (\uparrow p \cup \uparrow q \cup \downarrow p \cup \downarrow q) \cap (\uparrow(\downarrow p \cap \downarrow q) \cap \downarrow(\uparrow p \cap \uparrow q)).$$

We do not know of the further significance of this at this time, and in particular what the meaning of these kinds of expressions are specifically in the context of $\mathcal{P}$ being a CL $\mathcal{L}$. Indeed, we are suspicious that we are probably recapitulating or generalizing known results from lattice theory [3].

Ontology Induction: One of the great challenges in ontology work is the ability to create ontologies from other information sources such as relational or statistical data. CLs provide such an opportunity. In our case, we are working with molecular biologists and machine learning researchers who are creating relational knowledge bases of the interaction between a set of proteins $R = \{r_i\}$ and ligands (smaller molecules which bind to them to form biologically active complexes) $L = \{l_j\}$. As illustrated in Fig. 7, this provides a formal context in $R \times L$ relating proteins to those ligands with which they bind, and the resulting CL is a semantic hierarchy categorizing proteins r in the context of those ligands l which they bind, and *vice versa.* In this way, an actual POSO $\mathcal{O}$ is generated, where concepts $P = 2^L$ are collections of ligands, $\leq$is $\subseteq$on *L*, $X = R$, and *F* is determined by the concept lattice. Thus

	l_1	l_2	l_3
r_1	1	0	1
r_2	0	1	0
r_n	1	1	0

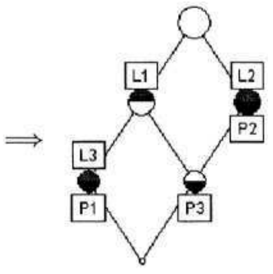


Fig. 7. Mapping a protein-ligand binding relation to its concept lattice proto-ontology.

$\mathcal{O}$ is fodder for our methodology, including POSOC for categorization, but also explorations of mappings from these proto-ontologies to other existing ontologies such as the GO or EC.

Anomaly Detection: We conclude with the final direction in which we would like to take this work, namely the use of measures in semantic hierarchies to detect anomalies in relational data as represented in CLs. Simply put, depending on the semantics of the formal context being represented, there may or may not be an expected distribution of nodes in the CL with respect to their cardinalities, that is $|A|$ and $|B|$. In other words, object concepts (where $|A| = 1$) should be "low" in the hierarchy, and attribute concepts (where $|B| = 1$) "high". When this is not the case, it indicates an usual object, attribute, or collection thereof. Much more needs to be explored here, but for now we will leave this as a suggestion for the community to consider further development.

Acknowledgements. Many people supported the original development of GOC and POSOC, especially Susan Mniszewski at LANL and Andy Fulmer and Gary Heaton at P&G. Recent support has been from the Protein Function Inference project at LANL, and in particular from Tom Terwilliger of LANL Biosciences and Michael Wall of LANL Computer Sciences.

References

1. Aigner, M: (1979) *Combinatorial Theory,* Springer-Verlag, Berlin
2. PG Baker, CA Goble, S Bechhofer, N Paton, R Stevens, A Brass: (1999) "An Ontology for Bioninformatics Applications", *Bioinformatics,* v. **15**:6, pp. 510-520
3. Birkhoff, Garrett: (1940) *Lattice Theory,* v. **25**, AMS, Providence RI, 3rd edition
4. Davis, Anthony R: (2000) *Types and Constraints for Lexical Semantics and Linking,* Cambridge UP
5. B Ganter and W Rudolf: (1999) *Formal Concept Analysis,* Springer-Verlag
6. Gene Ontology Consortium: (2000) "Gene Ontology: Tool For the Unification of Biology", *Nature Genetics,* v. **25**:1, pp. 25-29
7. Joslyn, Cliff: (1996) "Aggregation and Completion of Random Sets with Distributional Fuzzy Measures", *Int. J. of Uncertainty, Fuzziness, and Knowledge-Based Systems,* v. **4**:4, pp. 307-329
8. CA Joslyn and SM Mniszeiski: (2002) "DEEP: Data Exploration through Extension and Projection", LAUR 02-1330, `ftp://wwwc3.lanl.gov/pub/users/joslyn/deep.pdf`
9. CA Joslyn, SM Mniszewski, AW Fulmer, GA Heaton: (2003) "Measures on Ontological Spaces of Biological Function", *Pacific Symp. Biocompuating (PSB 03)*
10. Joslyn, Cliff; Mniszewski, Susan; Fulmer, Andy, and Gary Heaton: (2003) "Structural Classification in the Gene Ontology", in: *Proc. 6th Bio-Ontologies Workshop, Intelligent Systems for Molecular Biology (ISMB 03)*
11. Joslyn, Cliff and Mniszewski, Susan: (2004) "Combinatorial Approaches to Bio-Ontology Management with Large Partially Ordered Sets", in: *SIAM Workshop on Combinatorial Scientific Computing (CSC 04)*
12. Joslyn, Cliff; Mniszewski, Susan; Fulmer, Andy, and Gary Heaton: (2004) "The Gene Ontology Categorizer", submitted to the *2004 Conf. on Intelligent Systems for Molecular Biology (ISMB 04)*

13. Klir, George and Doug Elias: (2003) *Architecture of Systems Problem Solving,* Plenum, New York, 2nd edition
14. Knoblock, Todd B and Rehof, Jakob: (2000) "Type Elaboration and Subtype Completion for Java Bytecode", in: *Proc. 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*
15. Malvestuto, FM: (1996) "Testing Implication of Hierarchical Log-Linear Models for Probability Distributions", *Statistics and Computing,* v. **6**, pp. 169-176
16. Schröder, Bernd SW: (2003) *Ordered Sets,* Birkhauser, Boston
17. Sowa, John F: (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations,* Brooks/Cole, Pacific Grove
18. Verspoor, Karin; Simas, Tiago; Joslyn, C, *et al.:* (2004) "Protein Annotation as Term Categorization in the Gene Ontology", submitted to *BioCreative Workshop*

Concept Types and Coreference in Simple Conceptual Graphs

Michel Chein and Marie-Laure Mugnier

LIRMM - Université Montpellier 2 and CNRS
161, rue Ada - 34392 Montpellier cedex 5 - France
{chein,mugnier}@lirmm.fr

Abstract. This paper tackles the question of representing and reasoning with types and coreference in simple conceptual graphs (SGs). It presents a framework integrating a number of previous works. This proposal is guided by the usability of CGs in practice. In other words, notions should be easy to use in knowledge representation and operations for doing reasoning have to be efficiently realizable. We propose to use *conjunctive concept types,* which are conjunctions of primitive types. The conjunctive concept type set is defined by means of a primitive type set and a set of *banned* conjunctive types. For efficiency reasons our framework is based on projection. However it has been shown that projection is complete (w.r.t. logical deduction) only when SGs are in *normal form.* In some situations the original form of the SGs has to be kept; we thus define an extension of projection, called *coref-projection,* which is complete for SGs of any form. Coref-projection is in particular suitable for frameworks where it is not assumed that coreferent nodes are mergeable.

1 Introduction

This paper tackles the question of representing and reasoning with types and coreference in simple conceptual graphs (SGs). The analysis is guided by the usability of CGs in practice. More specifically the following points are emphasized:

- All notions should be *simple* to understand from a modeling viewpoint;
- Operations for doing reasoning should be realizable in an *efficient* way;
- Reasoning should be sound and *complete* w.r.t. to the formal semantics of SGs.

Let us first briefly introduce the problem. With efficiency in mind, the fundamental operation for doing reasoning with SGs is a graph homomorphism called *projection.* In SGs several concept nodes can refer to a same entity. They are said to be *coreferent.* When it has no coreferent concepts, a SG is said to be *normal.* On normal SGs projection is complete w.r.t. its usual formal semantics (its set semantic or its logical semantic Φ). But in general it is not. Several questions arise then. Can any pair of concepts be coreferent? This question is related to the interpretation of the concept type set. In which cases does a SG possess an equivalent normal SG? When a SG has no equivalent normal SG or when this

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 303–318, 2004.

latter is not computable in practice, is there an alternative to projection? Finally is the coreference notion really interesting for simple SGs? We shall study these questions and present answers found in the literature as well as a framework integrating a number of previous works.

Concept types. In [Sow84] the concept type set is a *lattice*. In later works and in the proposed standard for CGs [NCI98] it becomes a partially ordered set with a greatest element, thus a less constrained structure. Besides the structure an essential point is how the type hierarchy is interpreted. When a type is interpreted as a set of entities, the order over the hierarchy is interpreted as the inclusion. Then according to the terminology of [BHP+92] the greatest lower bound *(glb)* of two types, when it exists, can be *lattice-theoretically* or *order-theoretically* interpreted. In the order-theoretic interpretation there is nothing more than the interpretation of the subtype relation by set inclusion. With the lattice interpretation the interpretation of the glb of two types is furthermore interpreted as the intersection of these sets. E.g. let `Building` and `OldThing` be two incomparable types and let `HistoricLandmark` be their glb (Figure 1 left, example from [CCW97]). With both interpretations, every entity of type `HistoricLandmark` is also of types `Building` and `OldThing`. With the lattice interpretation every entity of both types `Building` and `OldThing` is necessarily of type `HistoricLandmark`. With the order-theoretic interpretation the glb of two types is simply a subtype, thus a subset of their intersection; an entity of both types `Building` and `OldThing` is not necessarily a `HistoricLandmark`.

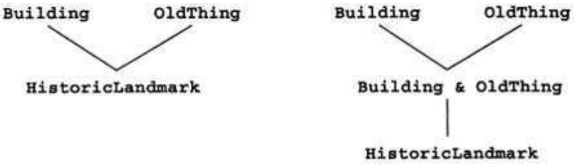


Fig. 1. Interpretation of types

The way the concept type set is interpreted has an effect on reasoning, as pointed out in [WL94] [CCW97]. For instance let Q be the query **`[HistoricLandmark:*]`** ("is there a HistoricLandmark?") and let G be an assertion containing the coreferent concept nodes `[Building:a]` and `[OldThing:a]` (thus G asserts that a is an entity which is a `Building` and an `OldThing`). With the lattice interpretation, G should provide an answer to Q while in the order-theoretic one it should not.

The lattice-theoretic interpretation requires to explicitely build all type intersections (for instance if a `Building` and `OldThing` is not necessarily a `HistoricLandmark` one has to build their common subtype `Building&OldThing` which is a supertype of `HistoricLandmark`, see Figure 1 right). Thus it leads to a number of new and artificial types which can be exponential in the number of useful types.

On the other hand, the drawback of the order-theoretic interpretation is that we loose the possibility of expressing conjunctions of types, e.g. that HistoricLandmark is exactly the intersection of OldThing and Building. This property could be expressed by a rule, stating that every entity of type Building and of type OldThing is also of type HistoricLandmark; however coding the conjunction of types in the hierarchy itself is more efficient than using rules, exactly as coding the subtype relation in a type hierarchy is more efficient than considering a flat set of types and representing the hierarchy by rules (cf. the discussion in [AKN86]).

A concept node represents an entity but an entity can be represented by several coreferent concept nodes. In order to ease the building and the understanding of a SG by a human being a way of gathering all or some concept nodes representing the same entity is desirable. A simple way to gather these nodes is to merge them into one node. This operation has to keep the meaning of the original SG. Thus if an entity is represented by a node of type t and by a node of type t' there must be a type representing exactly the entities which are of both types t and t'. More generally, we need to express that a type is the conjunction of several types. In [Bag01] a mechanism for type conjunction in conceptual graphs is proposed (see also [CCW97] where type conjunction is used in the context of fuzzy types). In Baget's framework, the set of conjunctive types is defined in intension by a hierarchy of primitive concept types. Any subset of (incomparable) primitive types defines a conjunctive type. The set of conjunctive types is partially ordered by an order extending that on primitive types.

However not all conjunctions of types have a meaning. Thus we need a way of expressing that the conjunction of two (or more) types is *banned,* or equivalently that they cannot have a common subtype. A classical way of doing is to add a special Absurd type below types with an empty intersection. But this technique is not always precise enough. For instance, t_1, t_2, t_3 being direct supertypes of Absurd means that the intersection of any two of them is empty. We cannot express that t_1 and t_2 are disjoint as well as t_1 and t_3, but that there can be an entity of type t_2 and t_3 (e.g. t_1 = **Animal**, t_2 = **Ship**, t_3 = **Robot**, all being subtypes of MobileEntity).

Giving in extension all acceptable types is not conceivable in practice, therefore we define the set of acceptable conjunctive types by the primitive type set and the (maximal) banned conjunctive types. In previous example, the types $\{t_1, t_2\}$ and $\{t_1, t_3\}$ would be banned. Theoretically, the number of banned conjunctive types can be exponential in the number of primitive types, but, practically, considering banned types of cardinality two seem to be sufficient and their number is at most quadratic in the number of primitive types.

Coreference. Coreference of several concept nodes can be expressed by the fact that these nodes have the same individual marker or by coreference links [Sow84]. Let us say that these nodes are in the same coreference set.

More or less restrictive definitions of the coreference notion can be found in the literature. These definitions are related to the interpretation of types and markers. Let us point out that we study here coreference in SGs, which is not the same relation as coreference in general CGs (for instance, coreference is a transitive relation in SGs whereas it may be not in general CGs). An individual marker is generally considered as a surrogate or an identifier (in the programming meaning) of an entity. According to the Unique Name Assumption usually made in knowledge representation, it is thus assumed that two distinct individual markers represent distinct entities. Hence nodes with distinct individual markers cannot belong to the same coreference set (once again the situation is not the same in general CGs, where for instance coreferent nodes with distinct individual markers a and b can be used to express that "a is not b"). An exception is the framework of [Dau03] which allows coreference between different individual concepts[1].

Let us consider two concepts with incomparable types t_1 and t_2. Can these nodes be coreferent without any condition on t_1 and t_2? In a lot of works the answer is positive. As already said, in our opinion important properties to ensure are, firstly, that *(a)* coreferent concepts can always be merged into a single node, secondly, that *(b)* the meaning of the obtained graph is the same as the one of the original one. With a lattice-interpretation of concept types these properties are ensured if the types of the coreferent nodes have a glb (distinct from the `Absurd` type), which becomes the type of the new node. With the order-theoretic assumption, and in absence of conjunctive types, the only way to ensure these properties is to impose that the set of types of the coreferent nodes possesses a minimal element in this set itself. Indeed suppose for instance there are two coreferent concepts with types `Building` and `OldThing` respectively. Either these concepts cannot be merged and *(a)* is not satisfied, or they can and the type of the new node is a subtype of `Building` and `OldThing` and the obtained graph is (in general) strictly more specialized than the original one. That is why in works where properties *(a)* and *(b)* are required it is imposed that coreferent nodes have the same type (e.g. [CMS98] [Ker01]). In the framework of conjunctive types proposed here *(a)* and *(b)* are obtained as soon as the conjunction of the types of the coreferent nodes is not a banned type. Then every SG can be easily transformed into an equivalent normal SG, which is called its *normal form.*

Whether a coreference set may contain both generic and individual nodes is a less important point. Indeed making a generic node and an individual node coreferent can always be performed by restricting the marker of the generic concept to the individual marker of the other node without changing the meaning of the graph.

In *general CGs* or in *positive nested CGs* coreference is mandatory to express identity of nodes belonging to different contexts. Now, is coreference a useful notion for SGs? In frameworks where it is not always possible to merge coreferent nodes of a SG the answer is obviously positive. But as soon as every SG possesses a normal form one may ask whether coreference is needed at all. Firstly, there is no doubt that coreference links are interesting in *user interfaces.* Indeed an

[1] This case corresponds to the existence of aliases among the individual markers. It could be included in our framework with slight modifications.

interest of SGs is their readability, quality that is lost when the SG becomes too big (e.g. too big to be entirely drawn on a screen). A SG too big can be split into pieces each displayable on one screen. The split can also be performed to decompose a SG into semantic modules, or to enhance a particular substructure. During a knowledge acquisition step a user can build a SG piece by piece, then connect them with coreference links. The same situation occurs when a SG basis has been constructed by different people or at different times. Secondly, even if the normal form can be computed in polynomial time, the variable of this polynomial is the size of the SG basis. Therefore, if the SG basis is very large (e.g. it is a significative part of the web or a document basis with millions of items) the cost of normalization can be prohibitive. Normalization can even be unthinkable if the basis is distributed among several servers.

Reasoning. Our framework is based on projection. However it has been shown that projection is complete only when SGs are in normal form. As explained above, in some situations one has to keep the original form of SGs; we thus propose an extension of projection, called *coref-projection,* which is complete without restrictions on the SG form. Coref-projection is in particular suitable for frameworks where it is not assumed that coreferent nodes are mergeable.

Paper organization. The sequel of the paper is organized as follows. Section 2 is devoted to the representation of the concept type set and the coreference relation. We provide basic definitions and properties and identify fundamental algorithmic problems. Section 3 deals with reasoning. It introduces the notion of coref-projection and elicits the relationships between projection and coref-projection.

2 Concept Types and Coreference

2.1 Concept Types

Throughout this paper, notations of [DP02] are used for the notions about ordered sets. The first notion for dealing with concept types is the primitive concept type set.

Definition 1 (Primitive concept type set). *A* primitive concept type set *is a (finite) ordered set of primitive concept types. It is denoted by T. It possesses a greatest element, called the universal type, and denoted by* $\top$.

A conjunctive concept type is a set of primitive concept types. It can be *acceptable* or *banned.* A type is banned if there cannot be an entity of this type, conversely there can be entities of an acceptable type. If there are two comparable types, say $t_i \leq t_j$, in the acceptable conjunctive type $t = \{t_1, ..., t_n\}$, then the conjunction of t_i and t_j has the same meaning as t_i since every entity of type t_i is of type t_j. Therefore t_j is useless. In the same way, if the conjunctive type $t = \{t_1, ..., t_n\}$ is banned, and if $t_i \leq t_j$, then the smaller one, i.e. t_i is useless.

In order theory a subset of non comparable elements of an ordered set is called an *antichain.* Thus, the set of conjunctive types is the set of antichains of the primitive type set *T.*

Definition 2 (Conjunctive concept type). *A* conjunctive concept type *is given by a (non empty) set of non comparable types i.e. an antichain of T.*

A primitive type can be seen as a conjunctive type with one type. Conjunctive types are provided with a natural partial order which extends that on primitive types: a conjunctive type t is a specialization of a conjunctive type s if every primitive type of s has a specialization (possibly equal to itself) in t.

Definition 3 (Conjunctive concept type set). *Let T be the set of primitive concept types.* $T^{\sqcap}$ *denotes the set of all conjunctive types over T. It is provided with the following partial order, which extends the partial order on T: given two types* $t = \{t_1 \ldots t_n\}$ *and* $s = \{s_1 \ldots s_p\}$, $t \leq s$ *if for every* $s_j \in s$, $1 \leq j \leq p$, *there is a* $t_i \in t$, $1 \leq i \leq n$, *such that* $t_i \leq s_j$.

Proposition 1. $(T^{\sqcap}, \ \leq)$ *is a lattice.*

A direct proof of this property is straightforward: the set of minimal elements of the union of two antichains is the glb of these antichains, therefore $(T^{\sqcap}, \ \leq)$ is a join-semi-lattice, and since it has a maximal element, it is a lattice[2].

The property of being a banned conjunctive type is hereditary: if a conjunctive type *A* is banned, all conjunctive types less than *A* are also banned. Therefore, the banned types form a down-set and one can assume that only the maximal banned conjunctive types are given.

Definition 4 (Banned type set). *Let* $\mathcal{B}$ *denote a subset of non comparable conjunctive types. An element of* $T^{\sqcap}$ *is said to be* banned *w.r.t.* $\mathcal{B}$ *if it is less or equal to an element of* $\mathcal{B}$. $\downarrow \mathcal{B}$ *denotes the set of all banned types i.e.* $\downarrow \mathcal{B} = \{t \in T^{\sqcap} \mid \exists t' \in \mathcal{B}, t \leq t'\}$. *An element of* $\mathcal{B}$ *is a* basic *banned type.*

Banned types should not be confused with *negated* types. For instance, banned types do not allow to express that an entity is of type $\neg(t_1 \wedge t_2)$. Introducing a banned type, say $\{t_1, \ t_2\}$, is simply equivalent to adding the constraint "$\neg\exists x(t_1(x) \wedge t_2(x))$" to the vocabulary.

The set of acceptable conjunctive types can be exponentially bigger than the primitive type set. That is why the concept type hierarchy is not defined in extension but by means of the primitive type set and a set of assertions stating which types are banned. The set of acceptable types is obtained from $T^{\sqcap}$ by removing the banned types $\downarrow \mathcal{B}$.

[2] It is well-known that the set of antichains of an ordered set is a lattice. But, usually, the order relation considered is the "complement-dual" of ours i.e. $t \leq s$ if for every $t_i \in t$ there is a $s_j \in s$ such that $t_i \leq s_j$ (with this order the lattice of antichains is isomorphic to the lattice of down-sets of *T* whereas, in our case, it is isomorphic to the lattice of up-sets of *T*).

Definition 5 (Concept type hierarchy). *A* concept type hierarchy $\mathcal{T}_C$ *is given by a couple* $(T, \mathcal{B})$ *where:*

- T *is the set of primitive concept types*
- $\mathcal{B}$ *is the set of basic banned conjunctive types*
- $\mathcal{B}$ *complies with* T, *i.e. for all* $b \in \mathcal{B}$, *there is no type* $t \in T$ *with* $t \leq b$ *i.e.* $\downarrow \mathcal{B} \cap T = \emptyset$.

$\mathcal{T}_C$ *is defined as the set* $T^{\sqcap} \setminus \downarrow \mathcal{B}$. $T^{\sqcap}$ *is thus partitioned into the acceptable types* $\mathcal{T}_C$ *and the banned types* $\downarrow \mathcal{B}$.

The least upper bound *(lub)* in $T^{\sqcap}$ of two acceptable conjunctive types is acceptable, therefore:

Proposition 2. *The set of acceptable conjunctive types* $\mathcal{T}_C$ *is a meet-semi-lattice.*

Type definitions. Let us come back to the example of the introduction. We can now express that an entity is of type `Building` and `OldThing`. We could associate a very simple type definition mechanism authorizing to *name* conjunctions, such as `HistoricLandmark = {Building, OldThing}`. Then concept types could be labelled by conjunctions of primitive types and named conjunctions. One can always associate to such a type a conjunction of primitive types by recursively decomposing named conjunctions. Thus this mechanism can be added on top of the techniques proposed here without difficulties.

2.2 Concept Labels

CGs are defined with respect to a vocabulary, that we recall here to fix the notations.

Definition 6 (Vocabulary). *A* vocabulary *is a 3-tuple* $\mathcal{V} = (\mathcal{T}_C, \mathcal{T}_R, \mathcal{I})$.

- $\mathcal{T}_C$, $\mathcal{T}_R$ *and* $\mathcal{I}$ *are pairwise disjoint (finite) sets*
- $\mathcal{T}_C$ *is the* concept type hierarchy
- $\mathcal{T}_R$, *the* relation set, *is partially ordered by* $\leq$, *and is partitioned into subsets* $\mathcal{T}_R^1 \dots \mathcal{T}_R^k$ *of relations of arity* $1 \dots k$ *respectively* $(k \geq 1)$. *Two relations of distinct subsets are non comparable.*
- $\mathcal{I}$, *is the (finite) set of individual markers.* $*$ *being the generic marker, the set* $\mathcal{I} \cup \{*\}$ *is partially ordered in the following way:* $*$ *is the greatest element and elements of* $\mathcal{I}$ *are pairwise incomparable.*

Definition 7 (Concept label set). *The set of concept labels is the set of couples* (t, m) *such that* $t \in \mathcal{T}_C$ *and* $m \in \mathcal{I} \cup \{*\}$. *The* partial order on concept labels *is the product of the partial orders on* $\mathcal{T}_C$ *and* $\mathcal{I} \cup \{*\}$, *i.e.* $(t, m) \leq (t', m')$ *iff* $t \leq t'$ *and* $m \leq m'$.

Since the product of two meet-semi-lattices is a meet-semi-lattice, the following property holds.

Proposition 3. *The ordered set of concept labels is a meet-semi-lattice.*

Thus a set of labels possesses a lub, furthermore if it has at least a common lower label then it possesses a glb. When an entity is represented by several concepts we impose that the labels of these concepts satisfy two conditions. The first condition corresponds to the unique name assumption. The second condition requires that the conjunction of the types of the concepts representing an entity is not a banned type. These two conditions are gathered in the following definition.

Definition 8 (Compatible concept labels). *The concept labels* (t_1, m_1),- ... , (t_k, m_k) *are* compatible *if*

- $|\mathcal{I} \cap \{m_1, \ldots, m_k\}| \leq 1$
- *the conjunctive type* t *defined as the set of minimal primitive types of* $\cup_{i=1}^{k} t_i$ *is acceptable i.e. is in* T_C

Proposition 4. *The concept labels* (t_1, m_1), ... , (t_k, m_k) *are compatible iff they possess a greatest lower bound.*

The glb of concept labels (t_1, m_1), ... , (t_k, m_k) is the concept label (t, m) with $t = min(\cup_{i=1}^{k} t_i)$ and $m = min(\cup_{i=1}^{k}\{m_i\})$ i.e. m is the generic marker if all m_i are the generic marker, otherwise m is the unique individual marker appearing in the m_i. By extension, concepts are said to be compatible if their labels are compatible.

Definition 9 (Concept merging). *A set of concepts* $\{c_1, ..., c_n\}$ *is compatible if the set of their labels is compatible. Let* l *be the greatest lower bound of their labels. Merging* $c_1, ..., c_n$ *consists in replacing them by a unique node* c *with label* l.

2.3 Algorithmic Problems Related to Types

Let us now list the algorithmic problems to be solved.

Pb0. Compare two elements of an ordered set T

This problem occurs when one has to compare two primitive concept types (or two relations). A lot of different coding schemes of ordered sets have been proposed for this basic problem and for other problems over ordered sets (cf. [Fal98] [FJN91] [Thi01]). To simplify the presentation of the following problems we assume that Pb0 can be solved in $O(1)$ (otherwise the given complexities must be multiplied by the complexity of Pb0).

Pb1. Compute a conjunctive type
Data: an ordered set T, a subset $t \subseteq T$
Problem: compute the set of minimal elements of t i.e. $min(t) = \{t_i \in t | \forall t_j \in t, t_j \not< t_i\}$
Naïve algorithm: compare each element of t to the others; the complexity is $O(|t|^2)$.

Pb2. Compare two conjunctive types
Data: two antichains t and s of T
Question: does $t \geq s$ hold ?
Naïve algorithm: compare all elements of t to the elements of s; the complexity is $O(|t| \times |s|)$.
It may be necessary to check if $\mathcal{B}$, as given by a user, is an antichain of $T^{\sqcap}$, and that can be done by an algorithm for Pb2. Furthermore, if $\mathcal{B}$ is not an antichain it can be replaced by the set of maximal elements of $\mathcal{B}$, and that can be done by a variant of Pb1 (considering $T^{\sqcap}$ instead of T and *max* instead of *min*).

Pb3. Check the acceptability of a conjunctive type
Data: $T, \mathcal{B}$ and t an antichain of T
Question: is t acceptable w.r.t. $\mathcal{B}$, i.e. does $t \not\leq s$ hold for all $s \in \mathcal{B}$?
Naïve algorithm: compare, with an algorithm for Pb2, every element of $\mathcal{B}$ to t; the complexity is $O(size(\mathcal{B}) \times complexity(Pb2))$, where $\text{size}(\mathcal{B})$ is equal to the sum of the cardinalities of the elements of $\mathcal{B}$.
Remark: $\mathcal{B}$ complies with T iff any primitive type is acceptable, that can be checked in $O(size(\mathcal{B}) \times |T|)$.

Pb4. Compare two concept labels
Data: two concept labels (t, m) and (t', m')
Question: $(t, m) \leq (t', m')$?
Algorithm: compare t and t' with an algorithm for Pb2 (or Pb0 if the types are primitive); therefore, the complexity is the same as the one of Pb2 because the comparison of two markers is trivial.

Pb5. Check the compatibility of concept labels and compute their glb
Data: a set of concept labels $\{(t_1, m_1), \ldots, (t_k, m_k)\}$
Problem: is this set compatible ? If it is the case compute the glb of this set,
Algorithm: compute $t = min(\cup_{i=1}^{k} t_i)$ i.e. make the union of the t_i then use an algorithm for Pb1, check if t is acceptable (cf. Pb3), and if $\{m_1, \ldots, m_k\}$ has a glb m. If (t, m) exists it is the glb of the set of labels.

Open problem. As previously mentionned there are numerous coding schemes of ordered sets for efficiently solving Pb0. There are also coding schemes of lattices of antichains. But in order to solve very efficiently all preceding problems we need a scheme for coding *together* T, $\mathcal{B}$ and a dynamic set of antichains of T (e.g. the conjunctive types appearing in a SG).

2.4 Basic, Simple, and Normal CGs

A *basic conceptual graph (BG)* is a SG without explicit coreference links. It is required that, if an individual marker appears more than once in the BG, the conjunctive type associated with these occurences is an acceptable type.

Definition 10 (Basic Conceptual Graph). *A* basic conceptual graph *(in short BG) defined over a vocabulary $\mathcal{V} = (T_C, T_R, \mathcal{I})$, is a 4-tuple $G = (C, R, E, l)$ satisfying the following conditions:*

1. *G = (C, R, E) is a finite, undirected and bipartite multigraph called the* underlying graph *of G. C and R are the node sets, respectively of* concept nodes *and of* relation nodes. *E is the family of* edges.
2. *l is a labelling function of the nodes and edges of G which satisfies:*
   a) *A concept node c is labeled by a pair* $(\text{type}(c), \text{marker}(c))$, *where* type$(c)\in T_C$, *and* marker$(c)\in \mathcal{I} \cup \{*\}$.
   b) *For any individual marker i, the conjunctive type associated with the set of types of the concepts with marker i is in T_C (i.e. is not banned).*
   c) *A relation node r is labelled by an element $l(r)$ of T_R.*
   d) *The degree of a relation node r must be equal to the arity of $l(r)$.*
   e) *Edges incident to a relation node are totally ordered and they are labeled from 1 to the arity of $l(r)$). If there is an edge labeled i between the relation node r and the concept node c, this edge is denoted by (r, i, c).*

If an individual marker i appears more than once in a BG the types associated with i in concept labels have a glb (condition 2(b) of Def. 10). Thus these concepts are compatible. Therefore any BG, say *G,* can be transformed into a normal BG by merging concepts having the same individual marker. The result is unique, we call it the *normal form* of *G,* that we note *norm(G).* A BG has the same meaning as its normal form.

Having two concepts with the same individual marker is a particular case of coreference. In general coreferent nodes can be individual or generic. To indicate that a generic concept is coreferent with another concept a *coreference link* is used. It is classically represented in drawings by a dotted line between the two concepts. The coreference relation is the reflexive and transitive closure of the relation defined by coreference links and individual marker identity.

Note that the coreference relation is transitive in SGs, whereas it may be not in general CGs (depending on whether coreference is transitive or not, general CGs correspond to FOL *with* equality as in [BMT99] [Dau02] or *without* equality as in [Ker01]).

A SG is a BG provided with a coreference relation.

Definition 11 (Simple conceptual graph). *A* simple conceptual graph *(in short SG) over a vocabulary $\mathcal{V}$ is a 5-tuple $(C, R, E, l, coref)$ such that:*

- *(C, R, E, l) is a basic conceptual graph over $\mathcal{V}$*
- *coref is an equivalence relation over C such that:*
  - *the concepts of any coref class are compatible*
  - *if two individual concepts have the same marker then they are in the same coref class*

A BG can be seen as a SG with a *coref* relation completely determined by its individual concepts. A normal SG is such that *coref* is the identity relation i.e. each concept is coreferent only with itself. A normal SG can thus be seen as a BG. In other words *normal SGs* and *normal BGs* are the same objects.

Every SG, say *G,* possesses a normal form, which is obtained by merging all concepts belonging to a same coreference class. We note it *G/coref.*

Definition 12 *(G/coref). Let $G = (C, R, E, l, coref)$ be a SG. G/coref is the normal SG obtained as follows: for any coref class $X = \{c_1, \dots, c_k\}$ all the c_i are merged into one concept X. G/coref is also called the* normal form *of G, and denoted by norm(G).*

2.5 Algorithmic Problems Related to Coreference

Algorithms for solving the following problems can be built by using algorithms for the problems in section 2.3 and classical list structures for graphs (for each relation node, one has the ordered list of its neighbors; for each concept node c, the neighbor list is the list of couples (i, r) such that $(r,\ i,\ c)$ is an edge ; each concept node contains its label and its coref class; for each coref class one has the list of its elements, ...).

Pb6. Merge compatible concepts
Data: A SG G and a set of compatible concepts $\{c_1, ..., c_n\}$ of G.
Problem: Compute the SG obtained from G by merging $c_1, ..., c_n$.
Algorithm: first, compute l, the glb of their labels (cf. Pb5), then create a new concept node with label l, and with neighbor list the concatenation of the neighbor lists of the c_i (this last operation can be computed in linear time in the sum of the list sizes).

Pb7. Compute the normal form of a BG or a SG
Data: a BG or a SG G
Problem: compute the quotient graph $G/coref_G$
Algorithm: merge the nodes of each coreference class (cf. Pb6).

3 Reasoning

3.1 Reasoning on BGs

The fundamental notion for reasoning with SGs is the *specialization/generalization* relation. This relation over SGs is originally defined by means of a set of elementary graph rules but for *effective* computation on SGs a global operation is used, called projection. This operation is equivalent to elementary rules: G is a generalization of H (notation $G \geq H$) iff there is a projection from G to H (see [Mug00] for a synthesis on these notions).

Definition 13 (Projection). *Let G and H be two BGs. A* projection *from G to H is a mapping π from C_G to C_H and from R_G to R_H, which preserves edges and may specialize concept and relation labels, i.e.:*

1. $\forall(r,i,c) \in G, (\pi(r), i, \pi(c)) \in H$;
2. $\forall x \in C_G \cup R_G$, $l_H(\pi(x)) \leq l_G(x)$.

When there is a projection (say π) from G to H we say that G projects to H (by π). G is called the *source* graph and H is the *target* graph. The existence of a projection from G to H intuitively translates the fact that all information contained in G is also contained in H. But projection does not behave as expected on non normal graphs (as independently noticed in [GW95] and [CM95][MC96]). See Figure 2: G and H are intituively/logically equivalent but they are not for projection; indeed there is a projection from H to G but not from G to H.

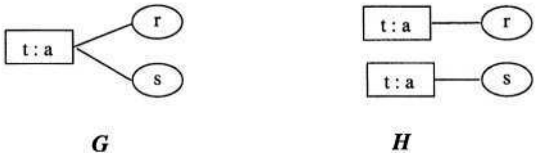


Fig. 2. Projection with a non normal target BG

Formally, projection is *sound* w.r.t. to the logical translation of BGs given by the mapping Φ of [Sow84] and *complete* when the target BG is in normal form.

Theorem 1 (Projection soundness and completeness). *Let G and H be two BGs defined on a vocabulary $\mathcal{V}$. Let $\Phi(\mathcal{V})$ be the logical translation of $\mathcal{V}$.*

- *If $G \geq H$ then $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$ [Sow84].*
- *If $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$ then $G \geq nf(H)$ [CM92] [CM95].*

3.2 Reasoning on SGs

Let us now consider SGs instead of BGs. A natural idea is to extend the BG projection in the following way: if two concepts are coreferent they must have coreferent images, that is to say either the same image (coref being a reflexive relation) or distinct coreferent images. We thus obtain the following definition.

Definition 14 (SG Projection). *Let $G = (C_G, R_G, E_G, l_G, coref_G)$ and $H = (C_H, R_H, E_H, l_H, coref_H)$ be two SGs. A* projection *π from G to H is a projection from the BG (C_G, R_G, E_G, l_G) to the BG (C_H, R_H, E_H, l_H) which preserves coreference i.e. $\forall x, y \in C_G, coref_G(x,y) \Rightarrow coref_H(\pi(x), \pi(y))$.*

As natural as it may seem previous definition does not solve the problem we have foreseen on BGs. Indeed, see that the SG projection behaves exactly as the BG projection in Figure 2. Let us study in more detail the relationships between SGs and their normal forms.

Proposition 5. *Every SG (thus every BG) projects to its normal form. The converse is false, as shown in Fig. 2 where G is the normal form ofH.*

Proposition 6. *Given SGs G and H, there is a bijective mapping between the set of projections from G to norm(H) and the set of projections from norm(G) to norm(H).*

The normal form of a SG always exists and is easily computable. However in some cases we have to do reasoning on SGs exactly as they are. An alternative to projection is then needed. The idea is that projection does not deal correctly with coreference because a concept cannot be mapped onto several concepts. If we modify this operation so that it maps coreference classes onto coreference classes we obtain the desired operation.

3.3 Coref-Projection

Definition 15 (Coref-projection). *Let* $G = (C_G, R_G, E_G, l_G, coref_G)$ *and* $H = (C_H, R_H, E_H, l_H, coref_H)$ *be two SGs. A* coref-projection *from G to H is a mapping* π *from* $coref_G$ *to* $coref_H$ *and from* R_G *to* R_H *such that:*

1. $\forall (r, i, c) \in G$, *let C be the coreference class of c, then there is a concept* $c' \in \pi(C)$ *such that* $(\pi(r), i, c') \in H$;
2. $\forall C \in coref_G$, *let* $l_G(C)$ *and* $l_H(\pi(C))$ *be the greatest lower bounds of the concept labels in C and* $\pi(C)$ *respectively; then* $l_G(C) \geq l_H(\pi(C))$;
3. $\forall r \in R_G$, $l_G(r) \geq l_H(\pi(r))$.

Each projection defines a coref-projection since by definition of a projection all coreferent nodes have their image in the same coreference class, but the converse is not true. The following property specifies the relationships between both notions.

Proposition 7. *Given SGs G and H, there is a bijective mapping between the set of coref-projections from G to H and the set of projections from norm(G) to norm(H).*

Since projection is (sound and) complete on normal SGs (theorem 1) we obtain the following corollary:

Corollary 1. *Coref-projection is sound and complete on SGs.*

Following property summarizes the "safe" ways of comparing two SGs.

Proposition 8. *Let G and H be two SGs. There is a bijective mapping between:*

- *coref-projections from G to H*
- *projections from norm(G) = G/coref to norm(H) = H/coref*
- *projections from G to norm(H).*

Let us also cite [CH94] which deals with BGs and, to obtain projection completeness, splits the source BG instead of computing the normal form of the target SG. With the idea of adapting this technique to SGs we could imagine to split the source SG into its so-called anti-normal form ([GW95]) in which a concept is incident to at most one edge. This method works well with a restricted definition of coreference (a coreference set does not mix generic and individual nodes and coreferent nodes have the same type) as presented in [Mug00]. Fig.3 shows that with the more general definition studied in this paper it does not work anymore: the antinormal form of G, denoted by *anf(G),* does not project to *H,* whereas G does project to *norm(H)* (actually G = *norm(H)).* Thus splitting the source SG into its anti-normal form is not an alternative to compacting the target SG into its normal form.

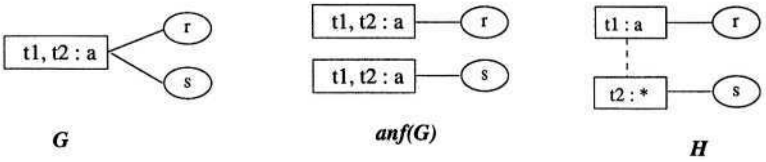


Fig. 3. Splitting the source SG is not an alternative

Equivalence with generalization/specialization rules. In same way as projection, coref-projection can be equivalently defined as a sequence of *generalization (or specialization)* rules. Because of space limitations we cannot detail these rule set here. Briefly, two new rules are added. These rules are "equivalence" rules in the sense that the output SG is equivalent to the input SG. The first rule merges two coreferent nodes into a single node, and the second one is the reciprocal rule that splits a node into two coreferent nodes (notice that if a concept c is splitted into coreferent nodes, say c_1 and c_2, the labels $l(c_1)$ and $l(c_2)$ are not necessarily equal to $l(c)$, but their glb is). Intuitively the first rule allows to go from a SG to its normal form. The rule set should also allow to generalize a SG by splitting a coreference class into two classes and reciprocally to specialize a SG by merging two coreference classes (provided that their union is a compatible set).

Finally, let us point out the interest of coref-projection in frameworks where it is not assumed that coreferent nodes can be merged into a single node (thus not all SGs have a normal form). See that the above definition of coref-projection can be reformulated replacing condition (2) by the following condition (2'): $\forall C \in coref_G$, $\forall c_i \in C$, $\exists c_j \in \Pi(C)$ with $l_G(c_i) \geq l_H(c_j)$. Equivalence between (2) and (2') follows from the definition of the partial order on conjunctive types[3]. Rewritten in this way coref-projection can be used without any assumption on type compatibility.

[3] Indeed, it is not true for any lattice that given two sets $E = \{a_1 ... a_p\}$ and $F = \{b_1 ... b_q\}$, $glb(E) \geq glb(F)$ implies for all a_i in E there is b_j in F with $a_i \geq b_j$.

4 Conclusion

In this paper we have proposed a framework integrating notions from previous works for dealing with the two related notions of conjunctive concept types and coreferent concept nodes in simple conceptual graphs. This framework keeps the qualities of CGs that we consider as essential. Namely, it is easy to use for knowledge representation because labelled graphs allow intuitive semantics based on drawings; reasoning is grounded on graph operations which are easily understandable; graph operations can be efficiently implemented; and last but not least they are sound and complete w.r.t. the usual FOL semantics.

In the same way as it is proposed for concepts, including conjunctive relations in SGs could also be done (following [Bag01]). Designing and implementing efficient algorithms is an important issue. In particular coref-projection checking is an NP-complete problem (since projection checking is [CM92]). Coref-projection being very close to projection, algorithmic techniques developed for projection (as in [Bag03]) should be adaptable to it.

References

[AKN86] H. Aït-Kaci and R. Nasr. Login: A logic programming language with built-in inheritance. *Journal of Logic Programming,* 3(3):185–215, 1986.

[Bag01] J.-F. Baget. *Représenter des connaissances et raisonner avec des hypergraphes: de la projection à la dérivation sous contraintes.* PhD thesis, Université Montpellier II, Nov. 2001.

[Bag03] J.-F. Baget. Simple conceptual graphs revisited: Hypergraphs and conjunctive types for efficient projection algorithms. In *Proc. of ICCS'03,* volume 2746 of *LNAI.* Springer, 2003.

[BHP$^+$92] C. Beierle, U. Hedtstück, U. Pletat, P. H. Schmitt, and J. H. Siekmann. An order-sorted logic for knowledge representation systems. *Artificial Intelligence,* 55(2):149–191, 1992.

[BMT99] F. Baader, R. Molitor, and S. Tobies. Tractable and Decidable Fragments of Conceptual Graphs. In *Proc. of ICCS'99,* volume 1640 of *LNAI,* pages 480–493. Springer, 1999.

[CCW97] T. H. Cao, P. N. Creasy, and V. Wuwongse. Fuzzy unification and resolution proof procedure for fuzzy conceptual graph programs. In *Proc. ICCS'97,* volume 1257 of *LNAI,* pages 386–400. Springer, 1997.

[CH94] B. Carbonneill and O. Haemmerlé. Rock : Un système question/réponse fondé sur le formalisme des graphes conceptuels. In *Actes du 9-ième congres RFIA,* pages 159–169, 1994.

[CM92] M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle,* 6(4):365–406, 1992. Available at http://www.lirmm.fr/~mugnier/.

[CM95] M. Chein and M.-L. Mugnier. Conceptual Graphs are also Graphs. Research Report RR-LIRMM 95-003, LIRMM, 1995. Available at http://www.lirmm.fr/~mugnier/.

[CMS98] M. Chein, M.-L. Mugnier, and G. Simonet. Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In *Proc. of KR'98,* pages 524–534. Morgan Kaufmann, 1998. Revised version available at http://www.lirmm.fr/~mugnier/.

[Dau02] F. Dau. *The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic.* PhD thesis, Technische Universitat Darmstadt, 2002.

[Dau03] F. Dau. Concept graphs without negations: Standardmodels and standardgraphs. In *Proc. ICCS'03,* volume 2746 of *LNAI.* Springer, 2003.

[DP02] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order.* Cambride University Press, Cambridge, UK, 2002.

[Fal98] A. Fall. The foundations of taxonomic encodings. *Computational Intelligence,* 14:598–642, 1998.

[FJN91] R. Freese, J. Ježek, and J.B. Nation. *Free Lattices.* American Mathematical Society, Providence, Rhode Island, 1991.

[GW95] B. C. Ghosh and V. Wuwongse. A Direct Proof Procedure for Definite Conceptual Graphs Programs. In *Proc. of ICCS'95,* volume 954 of *LNAI,* pages 158–172. Springer, 1995.

[Ker01] G. Kerdiles. *Saying it with Pictures: a logical landscape of conceptual graphs.* PhD thesis, Université Montpellier II and University of Amsterdam, Nov. 2001. Available at http://turing.wins.uva.nl/~kerdiles/.

[MC96] M.-L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle,* 10(1):7–56, 1996. Available at http://www.lirmm.fr/~mugnier/.

[Mug00] M.-L. Mugnier. Knowledge Representation and Reasoning based on Graph Homomorphism. In *Proc. ICCS'00,* volume 1867 of *LNAI,* pages 172–192. Springer, 2000. Revised version available at http://www.lirmm.fr/~mugnier/.

[NCI98] NCITS. Conceptual graphs: draft proposed American National Standard (dpANS). NCITS.T2/98-003; see also Sowa's web site "http://www.jfsowa.com", 1998.

[Sow84] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[Thi01] E. Thierry. *Sur quelques interactions entre structures de données et algorithmes efficaces.* PhD thesis, Université Montpellier II, Oct. 2001.

[WL94] M. Wermelinger and J.G. Lopes. Basic conceptual structures theory. In *Proc. ICCS'98,* volume 835 of *LNAI,* pages 144–159. Springer, 1994.

An Exportable CGIF Module from the CP Environment: A Pragmatic Approach

Heather D. Pfeiffer

Department of Computer Science
New Mexico State University
Las Cruces, New Mexico 88003-8001 USA
hdp@cs.nmsu.edu

Abstract. We have upgraded the Conceptual Programming Environment (CP) from a single standalone application to a set of component modules to increase flexibility of the environment and to allow any one of the modules to be used by applications outside of the environment. This allows the CP modules, in particular CGIF, to be tested pragmaticly in real applications. The CGIF module is encapsulated as a component and has been given an API specification. This module implements the NMSU modified ICCS2001 version of the CGIF interchange format for Conceptual Graphs (CGs) that is part of the dpANS and is setup to link with applications written in C, C++, Java^{TM} and Visual Basic 6.0. Communication is possible to all these languages by designing the API of the module so that it can accept either standard C string or Unicode string within the function calls. Since modules are flexible units of code, the components of the CP Environment have been tested for use under most versions of Microsoft Windows and different flavors of Linux.

1 Introduction

John Sowa developed Conceptual Structures (CS) [35,36] as a form of semantic modelling using first-order logic. These models are based on the works of C.S. Peirce [24] and represent declarative knowledge. This knowledge is implemented using connected multi-labeled bipartite oriented graphs that Sowa refers to as Conceptual Graphs (CGs). Each graph consists of nodes (with types - concept, relation, actor, specialcontext, comment) and edges (arcs connecting the nodes). There exists a mapping from each of these graphs to formulae in first-order logic.

In order for CGs to be transferred between working tools, either all the tools must be implemented within the same environment as a single application or there must be an interchange format. When tools are developed through a single system, the same data structure (or model) can be shared among all the tools so that data can be stored and retrieved. However, when tools are not part of the same system, they do not necessarily share the same internal data structure (or design model). To support interoperability for the CG-based applications [29], an interchange format referred to as "Conceptual Graph Interchange Format (CGIF)" has been proposed [37].

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 319–332, 2004.

This interchange format must be agreed upon by the whole working community. All though the CGIF document has been submitted as a standard, the final agreement for this format has not been reached within the community. Therefore, we present a pragmatic approach for enhancing and improving this format by offering a mechanism for interchange that can be placed in a testbed of applications. This is in response to the theoretical problems that are preventing the final agreement of this interchange format. We hope that as the community exchanges and uses CG s developed by different application tools, that the CGIF format will be enhanced, modified, and more clearly and precisely defined.

In this paper, we use the NMSU modified ICCS2001 version of CGIF[1]. At the "Workshop on Conceptual Graphs Tools (CGTools)" [28] at *ICCS2001,* at least part of the community agreed on this form of the interchange format. The major differences between this specification and the standard dpANS [40] are 1) CGStream has been added; 2) Qualifiers have zero or more quoted strings instead of **arcs***; 3) Special Contexts must have (no longer optional) a colon after the SpecialContextLabel before the definition; 4) Comments are just comments (no special types); 5) Referents begin with a colon and the colon is removed from the Concept definition; 6) All Type Definitions must have a colon between the type and label (same is true of Relation Definitions); and 7) The knowledge base is called a KB as oppose to a Module (this matches the original proposed standard [37]).

There are now several tools available within the CS community for processing Conceptual Graphs (CGs) and Concept Graphs (FCAs) for use with different applications. However, it is obvious that not all these applications use the same data structure for representing CGs and FCAs. Some of the data structures represent CGs as graphs with nodes and edges as in a semantic network structure [21]. Some examples are the Conceptual Programming Environment (CP) [25,26]; CGWorld [15,14]; GoGITaNT [3]; CGKEE [22]; and editors, such as, CharGer [11,10,1] and ARCEdit [27]. Other applications use concept lattices to define conceptual relationships as defined through Formal Concept Analysis (FCA) [17]. Some examples of these are ToscanaJ [18,6,5]; Docco [2]; and WebKB [23]. Other applications do not actually display CGs, but use the Conceptual Structure syntactic data and semantics to process information and define new languages; for example, pCG [8,7] and Prolog+CG [20]. Each of these applications has its own internal representation for CS and, therefore, they all use different data structures. One would not want to use the same data structure for all these systems as they have been optimized for either the application that is being implemented and/or the new languages that are being designed.

2 CP Environment

The Conceptual Programming Environment (CP) has been completely redesigned since originally presented by NMSU in 1992 [25,26]. Recalling that CP is a knowledge representation development environment with a graphical visualization framework, this set of tools uses graph structures and operations over those structures to do knowledge reasoning.

[1] http://www.cs.nmsu.edu/˜hdp/CGTools/cgstand/cgstandnmsu.html#Header_44 location

All knowledge within the system is stored and operated on as a graph. These graphs are implementations of Sowa's Conceptual Graphs [35], but also retain many of the features of graph theory [19]. Although there exists a mapping from CGs to formulae in first-order predicate calculus (FOPC), the operations used in the CP system take advantage of the graphical representation; therefore, the data structures and operations over the graphs use graph theory [19] instead of FOPC.

The original system was a single application written in Lisp and ran only on a Symbolics machine. The data structures were CGs defined using link lists of structure elements where the structures held the node information and the links were the edges of the graphs. All graphs had to be entered directly into the environment's editor, and each graph was stored into the environment's knowledge base. The CP inference engine would then operate over these data structures; sometimes creating new graphs or partial models of conceptual graphs and storing them into the environment's knowledge base. In the old environment, there was no way to import or export any of the graphs or models.

2.1 Motivation for Change

Over the last decade, we at NMSU have discovered that creating flexible, modularized environment is a much better design than a single application. Harry Delugach's invited talk at ICCS2003 [12] outlined a framework for building *active knowledge systems*. We agree with this view point and have designed the CP Environment to be the "heaven" displayed in his framework.

John Sowa, in a paper published in 2002 as part of a "Special Issue on Artificial Intelligence" of the *IBM Systems Journal* [38], proposed a modular framework as an architecture for intelligent systems because of the flexibility in communication and interoperability it provides. This flexible modular framework (FMF) allows different applications in different memory space to communicate using a blackboard architecture of message passing between applications. FMF would be very useful in implementing the reference framework discussed in Aldo de Moor's RENISYS specification methodology [13] because FMF handles interprocess communication across computers as well as processes, and it would also be useful in developing the intelligent agent operations from Delugach's framework [12]. However, the modularization of CP is at a module component level, rather than the FMF process communication level, so that the module can be directly "tied-in" to another application. The modular design at the component level also allows modules to be interchanged as units, as in modular furniture, to get the most flexibility from the environment.

The modularization of the CP Environment allows parts of the environment, the actual modules, to be both interfaced and interacted with by outside systems or applications. It also has a specific module, CGIF, that creates a mechanism to import and export CGs created from execution of the environment's inference engine modules and storage in the environment's knowledge base. This mechanism can be "plugged-in" to other applications by using the CGIF module's API specification to call the module's implementation code level [34]. Therefore, if an application currently available within the community does not have the ability to read and write CGIF format, this module could be "plugged-in" to the existing application to give it that functionality.

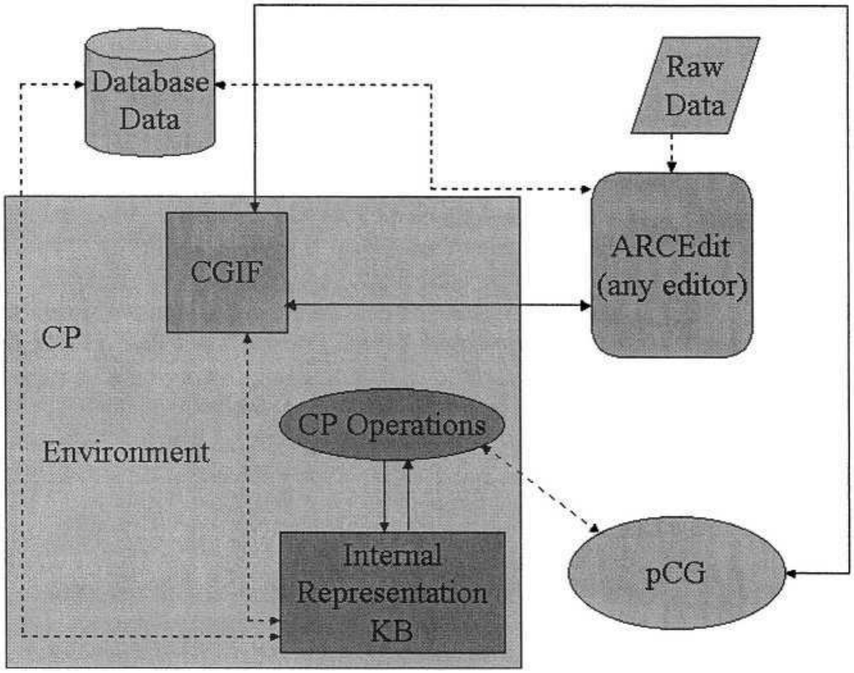


Fig. 1. Current CP Environment

2.2 Layout of Environment

Figure 1 depicts the new directionality of the CP Environment. The very light gray background area indicates what is actually part of the environment. The light gray oval depicts applications, i.e. the pCG reasoning and language system. The medium gray rounded-corner-square represents editors that are available for CGs, i.e. ARCEdit; these editors should be able to import/export CGIF formated files. The light gray trapezoid and drum shapes indicate data that is not necessarily graphical in nature, but may be part of a domain of information that a user wishes to process (note: the data in the database need not necessarily be textual and may be graphical or any visual form). The very dark gray shapes are modules that are part of the CP Environment and use the environment's internal data structures. All solid arrowed lines in the figure indicate data or processing that is currently available; dashed arrowed lines indicate where an interface, connection, interaction, and/or translation should be available between these elements, but is not currently present.

This newly designed environment has several areas of research that should be investigated. Some of these would be: data structures for syntactic data processing; database record processing; raw data processing; storage and retrieval of internal KB data processing; and reasoning and query processing. These areas should handle not only declarative and assumption knowledge, but also procedural knowledge containing time, space and other constraint information.

However, in this paper, we would like to look more closely at data structures for syntactic data processing. In particular, efficient and flexible ways of dealing with moving syntactic data information around (such as CGIF formatted graphs), both inside and outside of the CP environment.

3 Implementation Language Evaluation

Each of the applications and semantic languages defined in Section 1 may or may not share the same implementation language. So an added complication, besides different internal data structures, is that an application may not be able to communicate at a function call level with another application because they are not written in the same implementation language.

When we began to redesign the CP Environment, the question arose what implementation language should be used with the new modules. We examined first the CS editors that were currently available. These editors, for editing CGs and FCAs, have different implementation languages. CharGer is based on the API/Implementation code of Notio [34,33] which is written in Java^{TM}.ARCEdit is a plug-in to PowerPoint and is written in Visual Basic 6.0. ToscanaJ has an editor as part of its suite of tools written in Java^{TM}. While Docco is actually based on a Conceptual Email Manager [9], written in C++/QT, the commercial version of the manager [16] is a plug-in for Microsoft Outlook.

Since Notio, written in Java^{TM}, is already defined as an API/Implementation code level and is available with extendable class definitions, we considered using the Notio interface for the CGIF module. However, there are some drawbacks in communications to Java^{TM}(see Section 3.2), and Notio is in hiatus and is not currently being enhanced or developed [32].

All of the applications discussed in Section 1 employ different implementation languages, such as Prolog, XML, Schema, RDF, etc., which make it difficult to pass even simple syntactic representation data by linking languages in modules. Files, streams, pipes, blackboards, etc. can be used to pass data information without passing the actual data structures, but these mechanisms can be slow if there are a large number of graphs or the graphs are extremely complex. Every time one application process needs to talk to another, these mechanisms require multiple file descriptors to be opened. If the applications or systems execute on different machines, the FMF architecture is a flexible way of passing the syntactic data representation; but, if the applications and systems are able to be executed on the same machine configuration, a good API/Implementation design would be more advisable because the module can be linked directly into the existing application. Communication by files and other stream devices may require a locking mechanism to be setup, so that one application can know when it is safe to read the input graphs from another application. The locking of records can cause a problem when two applications communicate by way of shared databases or message passing systems, such as MPI. If on the other hand, an application/system can call another application/system directly (or can link to it), processing can go more quickly.

However, connecting systems when the implementation languages are not the same is more difficult, because a straight forward "call" to the other system's functions is not always possible. Each language implementation has its own calling specifications. In order to know which language would be best for implementation of the new environment's modular components, so that they could possibly be used directly by other applications, we performed an evaluation of how the C++ language interacts, interfaces and communicates with other languages.

3.1 C++ to C

Interfacing implementation code between languages that are somewhat similar, .i.e. C++ and C, is not as difficult as other communications between languages. However, this connection may not be bidirectional. The calling sequence for the C language is simpler than for the C++ language, because C++ does name mangling with the name of the function, the types of the arguments and the return type of the function. C does not do the same name mangling and uses only a modified form of the actual name of the function.

Therefore when designing an API in C++, the interface routines should be exported as "C" functions as opposed to being methods for a class in C++; this will prevent the routines from being name mangled by the C++ compiler. Wrapping C++ with standard C routines allows the internal implementation of the module to remain C++ and use the classes and methods functionality from C++, while at the same time using the simpler formulation of the name of the calling routine provided by C.

3.2 C++ to Java^{TM}

Connecting C++ to Java^{TM} is also possible, but is more difficult than communicating with C. This connection is also not bidirectional, but for different reasons. Java^{TM}is a simpler language than C++ [30], but it is an interpreted language. This means that Java^{TM}can be byte-compiled, creating a smaller file to be moved across the web, but it is not compiled to machine code. This allows Java^{TM}to be platform independent; C++ is a compiled language and is both platform dependent and operating system (OS) dependent. However, because Java^{TM}is interpreted instead of compiled, it can not be linked and called directly by a system (application) that is not written in Java^{TM}. Java^{TM} must start the process, and then can call compiled code in some of the other compiled languages (i.e. C/C++). Therefore, Java^{TM}can call interface functions written in C or C++, but C++ can not call Java^{TM} directly.

3.3 C++ to Visual Basic 6.0

The connection or interface from C++ to Visual Basic 6.0 is the most difficult connection among the four languages discussed in this paper. One reason is that Visual Basic 6.0 is a two part language; an event driven module part and a class module part.

The class module part is very similar to C++ and holds the characteristics that are available in object-oriented languages. Class modules can also be compiled just like C++ to native code for the machine. However, the event driven part executes Basic code in response to an event. These event driven procedures (routines) are triggered by a form or control which is hooked into the visual part of the language. The triggering of a routine by an event is similar to the interpretation of a function call in languages like Java^{TM}. Because of the event driven part of the language, Visual Basic 6.0 can call C++ or C API/Interface routines, but C++/C cannot trigger an event within the Visual Basic code, so the event procedures (routines) are not executed outside of Visual Basic code.

A second reason Visual Basic 6.0 is difficult to connect, is that it has different encoding of some of its data types than C, C++, or Java^{TM} [31]. Character data is stored in more bits by Visual Basic than by C. Therefore, to pass a character string as a parameter from Visual Basic to C or visa versa, the character string must be converted to Unicode first, that is passed as a parameter, and then decoded from Unicode at the other end. This makes passing character data much more cumbersome. Also, Visual Basic defines different boolean values than C; the "false" value is 0 in both languages, but the "true" value for Visual Basic is -1 (negative) where in C it is 1 (positive). Therefore, in passing boolean values, the user must be careful when working with conditionals.

3.4 Visual Basic .Net

If we were willing to have the CP Environment component modules available only for execution under the Microsoft Windows OS, Visual Basic .Net does not have any of the connection or interface problems discussed above. However, this would not allow the components implementation to be moved off the Microsoft Windows OS. If the CP modules are not implemented in Visual Basic .Net, than they can be made more widely available under Linux operating systems and eventually under other operating systems such as Unix.

4 API for CGIF Module

The CGIF module within the CP Environment is shown as the medium gray box labelled "CGIF" in Figure 1. The box is colored medium gray to depict that the functionality of this module was originally developed as part of the ARCEdit editor[27]. We decided that because the ARCEdit editor provides a way for graphs to be imported and exported to the CP Environment with a visual interface and able to import and export CGs in the CGIF format, we would remove the CGIF functions from ARCEdit and make a callable module that would become part of the environment. Weighing all learned during the evaluation of implementation languages discussed in Section 3, and wanting the new module to be portable to different operating systems, we selected the language C++ for implementing the new modules in the CP Environment.

By using C++ for the implementation of the CGIF component and defining an API to make it truly modular, we produced a DLL in the Microsoft Windows environment and later a C++ library in the Linux environment for the implementation of the module.

Definition 1. *CG*

> A conceptual graph is a list of zero or more concepts, conceptual relations, actors, special contexts, or comments.
>
> **CG ::= (Concept | Relation | Actor | SpecialContext | Comment)***
>
> The alternatives may occur in any order provided that any bound coreference label must occur later in the CGIF stream and must be within the scope of the defining label that has an identical identifier. The definition permits an empty CG, which contains nothing. An empty CG, which says nothing, is always true.

The API to the CGIF module was designed to allow: the creation of CGs as defined by the standard (see Definition 1 - from the CGIF standard document), the storage and retrieval of each of the CG elements (as in Definition 1), and the reading and writing of CGs (as in Definition 1) into a file in CGIF format for sharing among the user community. However, while the module's API was being designed, considerations were made to allow implementation languages beyond C and C++ to interface and interact with the module's implementation routines.

The API routines are defined to be standard C-named functions as wrappers around the C++ internal code. This is important to avoid any name mangling at compilation time and to allow other languages such as JAVA^{TM} to call the functions. The routines are also not attached to an event so they can be called by implementation languages other than Visual Basic 6.0. Using C++ instead of Visual Basic .Net allows the implementation of the module to be ported easily to operating systems other than Microsoft Windows. This was an added reason why the CGIF routines were not originally left in Visual Basic 6.0 and just made into a project by themselves (outside of ARCEdit). If this had been done, the CGIF module could have been made into a DDL for use on other Microsoft Windows machines, but could not be called by applications written in languages other than Visual Basic and could not have been ported to Linux.

As one can see in the Appendix, there are duplicates of some of the routines; this provides an interface for implementation languages, such as, Visual Basic 6.0 to pass character strings as arguments.

Example 1. Parse Concept (node) Routines

```
CGIF_API BOOLEAN STDCALL CGIF_ParseConcept( char *, BOOLEAN );
CGIF_API BOOLEAN STDCALL CGIF_ParseConceptW( BSTR, BOOLEAN);
```

An example of one of these paired routines can be seen in Example 1. In this example, the routines are almost exactly the same except that the second routine carries a 'W' after the routine name to indicate that the character string parameters or return arguments are in Unicode format, not standard C character strings.

Therefore when an implementation language needs to handle character strings as Unicode strings, they would call the routine from the API with the appended 'W'; C and C++ implementation routines can just call CGIF_ParseConcept directly.

Definition 2. *Concept*

> A concept begins with a left bracket"[" and an optional monadic type followed by optional coreference links and an optional referent in either order. It ends with an optional comment and a required "]".
>
> **Concept ::= "[" Type(1)? {CorefLinks?, Referent?} Comment? "]"**
>
> If the type is omitted, the default type is Entity. This rule permits the coreference labels to come before or after the referent. If the referent is a CG that contains bound labels that match a defining label on the current concept, the defining label must precede the referent.

The code within the two API routines is identical and both functions correctly implement the definition of a Concept as defined from the standard in Definition 2.

An example of using the paired routines seen in Example 1 would be if one parsed the phrase given in Example 2 using tools written in different languages.

Example 2. CG Concept Phrase
[CAT:'Felix']

If the ARCEdit editor was to parse the concept in Example 2, the second routine of the pair (CGIF_ParseConceptW) would need to be called, so that the string parameter would be sent in Unicode form. However, if the reasoning program pCG was to parse the Example 2 phrase, it would call the first routine (CGIF-ParseConcept) and use the C string type for the parameter. Both routines would create a concept node with the type: CAT and referent: 'Felix'.

5 Conclusion

Defining the CP environment to be both flexible and modular, makes it fit into the framework for beginning to build an *active knowledge system.* The modular design is at a component/unit level - rather than at the process/machine level - as seen in the FMF architecture. However, a module from the CP Environment allows other applications or systems to use a single module or a small group of modules from CP without requiring all of it.

The component module includes an API, so that other applications can directly link to it and need not incur the overhead of files, etc. The CP Environment design creates a way for a single module, i.e. CGIF, to be made available to other applications and systems written in different implementation languages, so that they can quickly incorporate it into their systems for handling the CGIF format. The CGIF module is available for download from the CP Environment's web page[2]. As other modules become available they can be downloaded, with source implementation code and complete documentation.

As the community begins to use the CGIF format to transfer Conceptual Graphs from tool to tool, benchmarks of graphs can begin to be built and used in testbeds, such as PORT

[2] http://port.semanticweb.org/CP/index.html

[4]. This will allow the user community to evaluate the features of each tool and how it might contribute to their research. Evaluation of the CGIF format would give different tool developers the opportunity to identify shortcomings in the current specifications and suggest possible theoretical design changes. The community's tool developers can then coorporate together to define more precisely the interchange format. By use of this pragmatic approach, we hope the CGIF format can be improved, and agreement might be reached within the community.

Within the CP Environment application, new modules can be added to translate CGIF format to other formats, such as the CLCE being developed currently by John Sowa [39]. This environment can be easily modified to use the FMF architecture by adding new component modules to communicate with the blackboard.

6 Future Research

Current research efforts for the CP Environment are to make the "CP Operations" module available to be linked by applications and systems outside the environment. This module has been encapsulated and an API is being defined. We will test the API/Implementation module by setting up communication between the CP Operations module and the pCG reasoning system. Once testing is complete, this module will be placed on the CP Environment web site [2] and made available to outside applications and systems. It should be noted, that the pCG reasoning system is implemented in Java^{TM}, but will be able to interface and interact directly with the CP Operations module, written in C++. The CP Operations module has the basic operations: maximal-join, project, generalize, and specialize. In the not so distant future, we hope that operations will be added to this module for performing operations over time, space and constraint processing.

Acknowledgements. Thanks to the many reviewers that took the time to help me improve and revise this paper. In particular, I would like to thank Mary Keeler for all her great comments and observations.

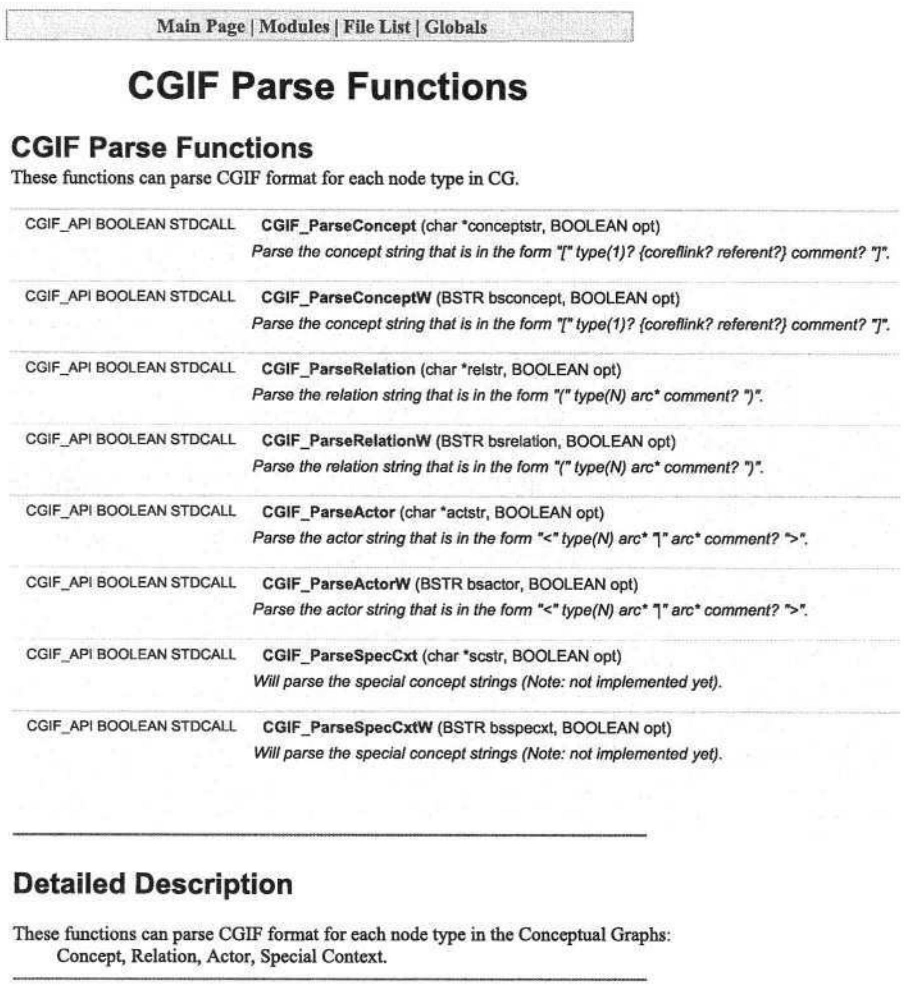
References

1. *CharGer - A Conceptual Graph Editor.* University of Alabama in Huntsville, Alabama, USA. http://www.cs.uah.edu/delugach/CharGer.
2. *Docco.* University of Queensland, Australia. http://tockit.sourceforge.net/docco/index.html.
3. *GoCITaNT.* LIRMM - Montpellier, France. http://cogitant.sourceforge.net/index.html.
4. PORT: The Peirce On-line Resource Testbed Project. http://port.semanticweb.org/.
5. Peter Becker. *ToscanaJ.* Technical University of Darmstadt, Germany. http://toscanaj.sourceforge.net/.
6. Peter Becker and Joachim Hereth Correia. The ToscanaJ suite for implementing Conceptual Information Systems. In G. Stumme, editor, *Formal Concept Analysis – State of the Art,* Berlin – Heidelberg – New York, 2004. Springer. To appear.
7. D. Benn and D. Corbett. An application of the process mechanism to a room allocation problem using the pCG language. In H.S. Delugach and G. Stumme, editors, *Conceptual Structures: Broadening the Base,* Springer-Verlag Lecture Notes in Computer Science 2120, pages 360–374, 2001.

8. D.J. Benn and D. Corbett. pCG: An implementation of the process mechanism and an extensible CG programming language. In *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001] URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
9. R.J. Cole, Peter Eklund, and Gerd Stumme. Document retrieval for email search and discovery using Formal Concept Analysis. *Applied Artificial Intelligence,* 17(3), 2003.
10. H. Delugach. CharGer: Some lessons learned and new directions. In G. Stumme, editor, *Working with Conceptual Structures - Contributions to ICCS 2000,* pages 306–309, 2000. Shaker-Verlag.
11. H. Delugach. CharGer: A graphical Conceptual Graph editor. In *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001] URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
12. H.S. Delugach. Towards building active knowledge systems with conceptual graphs. In A. de Moor, Wilfried Lex, and Bernhard Ganter, editors, *Conceptual Structures for Knowledge Creation and Communications,* pages 296–308, Heidelberg, 2003. Springer-Verlag. Lecture Notes in Artificial Intelligence, LNAI 2745.
13. A. deMoor. Applying conceptual graph theory to the user-driven specification of network information systems. In D. Lukose, H.S. Delugach, M. Keeler, L. Searle, and J.F. Sowa, editors, *Conceptual Structures: Fulfilling Peirce's Dream,* Springer-Verlag Lecture Notes in Artificial Intelligence 1257, pages 536–550. ICCS, Springer, August 1997.
14. P. Dobrev, A. Strupchaska, and K. Toutanova. CGWorld-2001: New features and new directions. In *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001] URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
15. P. Dobrev and K. Toutanova. CGWorld - a web based workbench for conceptual graphs management and applications. In G. Stumme, editor, *Working with Conceptual Structures - Contributions to ICCS 2000,* Shaker-Verlag, pages 243–256, 2000.
16. Email Analysis Pty Ltd, Australia. *Mail-Sleuth.* http://www.mail-sleuth.com/.
17. B. Ganter and R.Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer-Verlag, Berlin Heildelberg New York, 1999.
18. B. Groh, S. Strahringer, and R. Wille. TOSCANA-systems based on Thesauri. In Marie-Laure Mugnier and Michel Chein, editors, *Conceptual Structures: Theory, Tools, and Applications,* Springer-Verlag Lecture Notes in Artificial Intelligence 1453, pages 127–141, Heidelberg, August 1998. ICCS, Springer-Verlag.
19. F. Harary. *Graph Theory.* Addison-Wesley, Reading, MA, 1969.
20. A. Kabbaj and M. Janta-Polcynzki. From Prolog++ to Prolog+CG: A CG object-oriented logic programming language. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computional Issues,* pages 540–554, Berlin, 2000. Lecture Notes in Artificial Intelligence, vol. 1867, Springer-Verlag.
21. F. Lehmann, editor. *Semantics Networks.* Pergamon Press, Oxford, ENGLAND, 1992.
22. D. Lukose. CGKEE: Conceptual graph knowledge engineering environment. In Conceptual Structures: Fulfilling Peirce's Dream, editor, *Conceptual Structures: Fulfilling Peirce's Dream,* Springer-Verlag Lecture Notes in Artificial Intelligence 1257, pages 589–593. ICCS, Springer, August 1997.
23. P. Martin. The webKB set of tools: A common scheme for shared www, annotations, shared knowledge bases and information retrieval. In D. Lukose, H.S. Delugach, M. Keeler, L. Searle, and J.F. Sowa, editors, *Conceptual Structures: Fulfilling Peirce's Dream,* Springer-Verlag Lecture Notes in Artificial Intelligence 1257, pages 585–588. ICCS, Springer, August 1997.
24. C.S. Peirce. Manuscripts on existential graphs. *Peirce,* 4:320–410, 1960.

25. H.D. Pfeiffer and R.T. Hartley. The Conceptual Programming Environment, CP. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W. Ekland, editors, *Conceptual Structures: Current Research and Practice,* Ellis Horwood Workshops. Ellis Horwood, 1992.
26. H.D. Pfeiffer and R.T. Hartley. Temporal, spatial, and constraint handling in the conceptual programming environment, cp. *Journal for Experimental and Theoretical AI,* 4(2):167–182, 1992.
27. H.D. Pfeiffer and R.T. Hartley. ARCEdit - CG editor. In *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001]
URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
28. H.D. Pfeiffer and R.T. Hartley, editors. *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001]
URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
29. A. Puder. Mapping of CGIF to operational interfaces. In Marie-Laure Mugnier and Michel Chein, editors, *Conceptual Structures: Theory, Tools, andApplications,* Springer-Verlag Lecture Notes in Artificial Intelligence 1453, pages 119–126, Heidelberg, August 1998. ICCS, Springer-Verlag.
30. Kirk Radeck. C# and Java: Comparing programming languages. October 2003.
http://www.windowsfordevices.com/articles/AT2128742838.html.
31. Steven Roman. *Win32 API Programming with Visual Basic.* O'Reilly, first edition, 1999.
32. F. Southey. *NOTIO.* http://notio.lucubratio.org/index.html.
33. F. Southey. Notio and Ossa. In *CGTools Workshop Proceedings in connection with ICCS 2001,* Stanford, CA, 2001. [Online Access: July 2001]
URL:http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html.
34. F. Southey and J.G. Linders. NOTIO - a Java API for developing CG tools. In W. Tepfenhart and W. Cyre, editors, *Conceptual Structures: Standards and Practices,* pages 262–271, Berlin, 1999. Springer-Verlag. Lecture Notes in Artificial Intelligence, LNAI 1640.
35. J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, Reading, MA, 1984.
36. J.F. Sowa. Conceptual graphs as a universal knowledge representation. In F. Lehmann, editor, *Semantics Networks,* Oxford, ENGLAND, 1992.
37. J.F. Sowa. Conceptual graphs: Draft proposed american national standard. In Conceptual Structures: Standards and Practices, editors, *Conceptual Structures: Standards and Practices,* pages 1–65, Berlin, 1999. Springer-Verlag. Lecture Notes in Artificial Intelligence, LNAI 1640.
38. J.F. Sowa. Architectures for intelligent systems. In *Special Issue on Artificial Intelligence,* volume 41 of *IBM Systems Journal,* pages 331–349. 2002.
39. J.F. Sowa. Common Logic Controlled English.
http://www.jfsowa.com/clce/specs.htm, February 2004.
40. J.F. Sowa and et al. *Conceptual Graph Standard, American National Standard NCITS,* T2/ISO/JTC1/SC32 WG2 M 00 edition. [Access Online:April 2001],
URL: http://www.bestweb.net/ sowa/cg/cgstand.htm.

Appendix: Example Documentation



Main Page | Modules | File List | Globals

CGIF Parse Functions

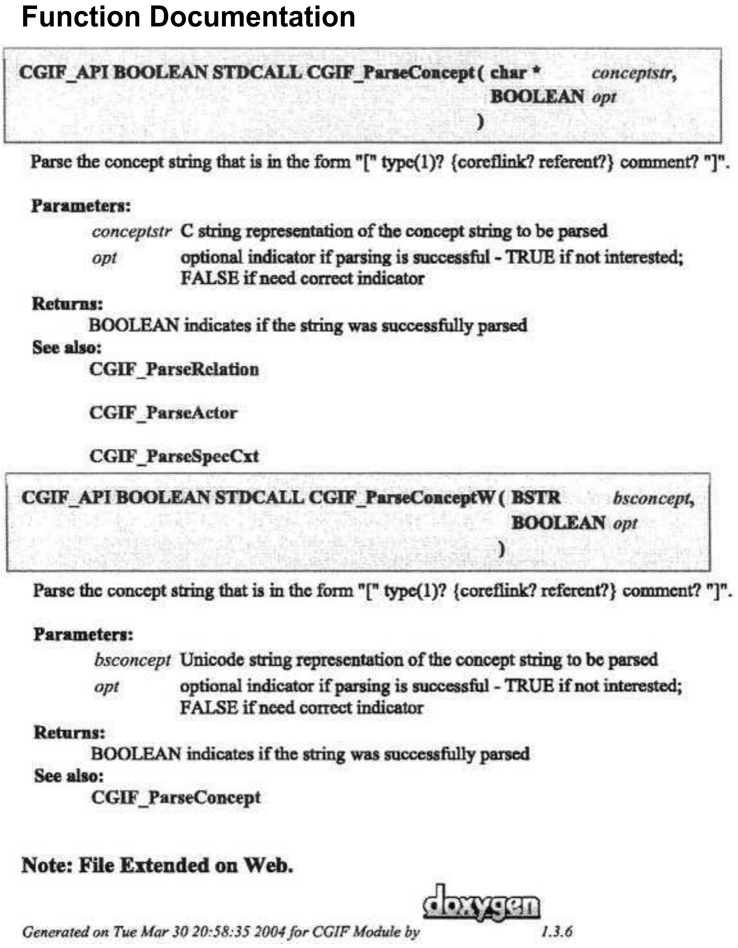
CGIF Parse Functions

These functions can parse CGIF format for each node type in CG.

CGIF_API BOOLEAN STDCALL	**CGIF_ParseConcept** (char *conceptstr, BOOLEAN opt) *Parse the concept string that is in the form "[" type(1)? {coreflink? referent?} comment? "]".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseConceptW** (BSTR bsconcept, BOOLEAN opt) *Parse the concept string that is in the form "[" type(1)? {coreflink? referent?} comment? "]".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseRelation** (char *relstr, BOOLEAN opt) *Parse the relation string that is in the form "(" type(N) arc* comment? ")".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseRelationW** (BSTR bsrelation, BOOLEAN opt) *Parse the relation string that is in the form "(" type(N) arc* comment? ")".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseActor** (char *actstr, BOOLEAN opt) *Parse the actor string that is in the form "<" type(N) arc* "|" arc* comment? ">".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseActorW** (BSTR bsactor, BOOLEAN opt) *Parse the actor string that is in the form "<" type(N) arc* "|" arc* comment? ">".*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseSpecCxt** (char *scstr, BOOLEAN opt) *Will parse the special concept strings (Note: not implemented yet).*
CGIF_API BOOLEAN STDCALL	**CGIF_ParseSpecCxtW** (BSTR bsspecxt, BOOLEAN opt) *Will parse the special concept strings (Note: not implemented yet).*

Detailed Description

These functions can parse CGIF format for each node type in the Conceptual Graphs:
Concept, Relation, Actor, Special Context.



Function Documentation

```
CGIF_API BOOLEAN STDCALL CGIF_ParseConcept( char *     conceptstr,
                                            BOOLEAN    opt
                                          )
```

Parse the concept string that is in the form "[" type(1)? {coreflink? referent?} comment? "]".

Parameters:

conceptstr	C string representation of the concept string to be parsed
opt	optional indicator if parsing is successful - TRUE if not interested; FALSE if need correct indicator

Returns:

BOOLEAN indicates if the string was successfully parsed

See also:

CGIF_ParseRelation

CGIF_ParseActor

CGIF_ParseSpecCxt

```
CGIF_API BOOLEAN STDCALL CGIF_ParseConceptW( BSTR      bsconcept,
                                             BOOLEAN   opt
                                           )
```

Parse the concept string that is in the form "[" type(1)? {coreflink? referent?} comment? "]".

Parameters:

bsconcept	Unicode string representation of the concept string to be parsed
opt	optional indicator if parsing is successful - TRUE if not interested; FALSE if need correct indicator

Returns:

BOOLEAN indicates if the string was successfully parsed

See also:

CGIF_ParseConcept

Note: File Extended on Web.

Generated on Tue Mar 30 20:58:35 2004 for CGIF Module by doxygen *1.3.6*

Using Conceptual Graphs to Represent Agent Semantic Constituents

Lois W. Harper and Harry S. Delugach

Department of Computer Science
N300 Technology Hall
University of Alabama in Huntsville
Huntsville, AL 35899 USA
{lharper, delugach}@cs.uah.edu

Abstract. This paper develops two agent knowledge bases in conceptual graph form, one using the KD45 underlying logical model for *belief* and one without any underlying logical model for *belief*. Action-attitudes in the knowledge bases provide contexts that represent the agents' mental attitude towards, and willingness to act upon information in the knowledge bases. Preconditions for communication acts are also represented in the knowledge bases as well as mental attitude changes following communications. Conceptual graphs are a flexible and extendable form of knowledge representation that is used to capture and represent semantic constituents of communications in a form that may be used by software agents. The knowledge base representations in this paper provide software agents a perspective from which they may reason about the communicating agent's *beliefs* and communication actions.

1 Introduction

Software agent technologies are being directed towards enabling heterogeneous computing platforms in open environments to communicate, share resources and cooperatively delegate tasks. Software agents provide developers a high-level abstracted view from which to more easily solve problems in these programming environments. [1] *Intelligent* software agents are often characterized as holding mental attitudes, such as *belief,* towards their knowledge. These agents may use an agent communication language (ACL), such as KQML or the FIPA ACL, to convey their mental attitudes towards communicated knowledge. [2, 3]

An underlying assumption with ACLs is that the communicating software agents 'speak' the same ACL. [4] This presents difficulties for software agents designed to converse in different ACLs, or to converse with different dialects of the same ACL. Other approaches have been taken towards establishing common vocabularies between software agents. Some examples are:

- [5] The World Wide Web Consortium (W3C) Ontology Web Language (OWL) with the Resource Description Framework (RDF) allows web-based ontologies to

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 333–345, 2004.

be established as meta-languages, providing web agents shared vocabularies. The DAML-S ontology supports locating web services. [6]

- Gmytrasiewicz et al.[7] proposes that software agents negotiate and establish a common vocabulary at runtime, while Reed et al. [8] propose that communicating agents share a common set of semantic primitives but negotiate the final meaning of these primitives at runtime.
- Sycara et al. [9] use matchmaker agents to support advertising, requesting and matching agent services using a communication language in addition to their ACL, called the Agent Capability Description Language (ACDL) LARKS. Matchmaker agents are not information brokers, so after LARKS is used to locate services, the matched agents must still intercommunicate to use services.

These approaches address agent vocabularies in different semantic constituents: ACLs for agent communication protocols; negotiated semantics for heterogeneous agent messages; matchmaker agents for locating agent services, and web ontologies for establishing and using web-based domain knowledge.

Once a common vocabulary of any type has been established, the expected use of this vocabulary in dynamic situations, such as communication protocols, needs to be represented to software agents. These representations may assist heterogeneous agents in reasoning about and overcoming other communication barriers. Hodgson lists these requirements for representation schemes in AI, emphasizing that the last point is most important [10]:

- The representation scheme should “permit sufficient internal organization so that the type of object (problem) being represented can be easily classified.”
- The representation scheme should “be flexible enough so that decompositions of problems can be expressed naturally in terms of the representation.”
- The representation scheme should “use as few special structures as possible.”

For example, when representing communication protocols in the FIPA ACL, the type of problem is a procedure, and the problem is expressed in terms of the agents’ mental attitudes towards knowledge, using the Kripke’s KD45 possible world semantics for the mental attitude *belief* [3]. Delugach has developed an extension to conceptual graphs to capture dynamic behaviors [11]. Sowa has discussed the use of context in conceptual graphs for modal logic representation [12, 13]. This suggests that conceptual graphs are a notational framework that will satisfy the representation requirements listed above when representing FIPA agent communications.

2 A Kripke Model of Belief in Conceptual Graph Form

Many theories of rational agency have been developed to allow developers to reason about the behaviors of *intelligent (rational)* agents. Four well known theories of rational agency, as described in [14, 15] are:

- Cohen and Levesque [16] – temporal logic + belief (KD45) +goal (KD)
- Moore [17] – dynamic logic + knowledge (S5)

- BDI [18] – temporal logic + belief (KD45) + desire (KD) + intention (KD)
- KARO [19] – dynamic logic + belief (KD45) + wishes (KD)

These theories use the KD45 modal logic model for *belief.* The FIPA ACL's logical model for *belief* is also the Kripke KD45 logical model.[3] Software agents that reason about communicating in the FIPA ACL may therefore benefit from access to a KD45 model of *belief.*

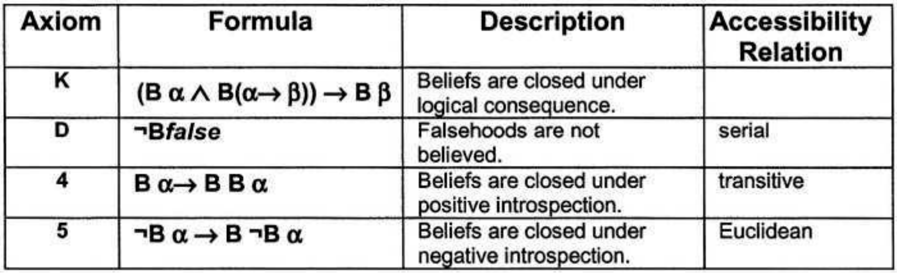
2.1 KD45 Modal Belief Logic

The following discussion is taken from [20, 21]. Modal *belief* logics typically add a modal operator for *belief,* often denoted as **B**, to first-order logic. The rule "If α is a well formed formula, then $\mathbf{B}\ \alpha$ is also a well formed formula" is also added.

A Kripke modal frame is a structure **M** = < **W**, **D**, **R**, **F**> where: **W** is a non-empty set of possible worlds; **D** is a non-empty domain of individuals; **R** is a binary accessibility relation on **W**; and **F** is a state dependent interpretation function. A formula α is satisfiable with respect to the modal structure if there exists a triple (M, w, f) such that $(M, w, f) \models \alpha$. A formula α is valid if for every triple (M, w, f) it is the case that $(M, w, f) \models \alpha$. For example, $(B\alpha \vee \neg B\alpha)$ is valid, and $(B\alpha \vee B\neg\alpha)$ is satisfiable but not valid. [20]

The term 'KD45' identifies four formulas for *belief,* shown in the following table. Kripke's insight was to show that these accessibility relations correspond to these formulas.

Table 1. KD45 Formulas and Corresponding Frame Relationships



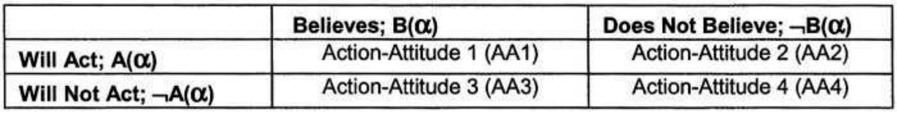
Axiom	Formula	Description	Accessibility Relation
K	$(B\ \alpha \wedge B(\alpha \rightarrow \beta)) \rightarrow B\ \beta$	Beliefs are closed under logical consequence.	
D	$\neg B$*false*	Falsehoods are not believed.	serial
4	$B\ \alpha \rightarrow B\ B\ \alpha$	Beliefs are closed under positive introspection.	transitive
5	$\neg B\ \alpha \rightarrow B\ \neg B\ \alpha$	Beliefs are closed under negative introspection.	Euclidean

2.2 Mental Attitudes and Action Attitudes

The FIPA specification states that software agents using the FIPA-ACL will possess the three primitive mental attitudes of *belief, uncertainty* and *choice,* formalized in the FIPA Semantic Language by the modal operators **B**, **U** and **C** respectively. An FIPA agent is *uncertain* (**U**) about a proposition *p* if it considers that *p* is more likely to be true than not true. The FIPA mental attitude *choice* (**C**) represents a goal state concerning a proposition.[3]

An autonomous agent may *believe* a proposition α, $B(\alpha)$, but may choose not to take action on that *belief.* For example, the agent may receive a query concerning the truth-value of a proposition that it *believes* is true, but may choose not to answer the query. As intelligent and autonomous software processes, software agents will make decisions based on their input, existing knowledge, interaction protocols and other established plans. [22-24] These factors are not directly addressed by an agent communication language, but nonetheless will contribute towards determining when and what communication acts may take place. The predicate variable **A,** for *action,* is used in this paper to represent these factors that, together with the mental attitude of *belief* differentiate four mutually exclusive "action-attitudes". These are shown in Table 2.

Table 2. Four Mutually Exclusive Action-Attitudes



	Believes; B(α)	Does Not Believe; ¬B(α)
Will Act; A(α)	Action-Attitude 1 (AA1)	Action-Attitude 2 (AA2)
Will Not Act; ¬A(α)	Action-Attitude 3 (AA3)	Action-Attitude 4 (AA4)

2.3 A Conceptual Graph Model of the Kripke KD45 Frame

In this section we develop a representation of the KD45 model of *belief* in conceptual graph form. The FIPA Communicative Act Library Specification does not specify any agent implementation. [3] Our model is simply developed for this example of showing how a model of *belief,* in a form accessible to software agents, may assist agents in reasoning about communication acts.

- Since the FIPA ACL uses the Kripke KD45 model of *belief,* we see that these action attitudes may correspond to possible worlds in a Kripke *belief* structure.
- Since communication acts may be initiated by agents that have no knowledge of the truth-value of a proposition, such as in the FIPA query-if performative, we let the mental attitude *choice,* **C**, be a subworld of AA2. [3]
- Because the preconditions for communication in the FIPA ACL ensure that no proposition is conveyed by an agent that is uncertain about the truth-value of the proposition, we let the mental attitude *uncertain,* **U**, be a subworld of AA3. [3]
- Because a speech act is an action according to the speech act theories of [25, 26], preconditioned by *beliefs* specified in [3], we do not use AA4 in our model.

A Kripke frame with three worlds, each an action-attitude described above, and with the KD45 accessibility relation is then:

- A set W whose elements are all possible worlds (AA1, AA2, AA3)
- The accessibility relations: R(AA1, AA2), R(AA2, AA1), R(AA2, AA3), R(AA3, AA2), R(AA3, AA1), R(AA1, AA3)

The KD45 Kripke frame in Figure 1 is represented in conceptual graph notation in Figure 2. Delugach has extended conceptual graph notation to represent dynamic behaviors, such as the assertion of temporal knowledge, using a relationship called a demon [11]. When activated, the demon will assert each of its output concepts and retract each of its input concepts. The demons in the conceptual graph below are activated by changes in action-attitudes held by the agent, with respect to a proposition. Within the context of possible world semantics, these demons allow an agent to change state between possible worlds. The previous world (action-attitude) is retracted with respect to a proposition, and a new world (action-attitude) is asserted.

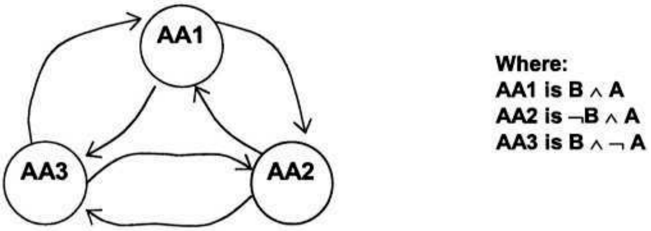


Fig. 1. KD45 Frame

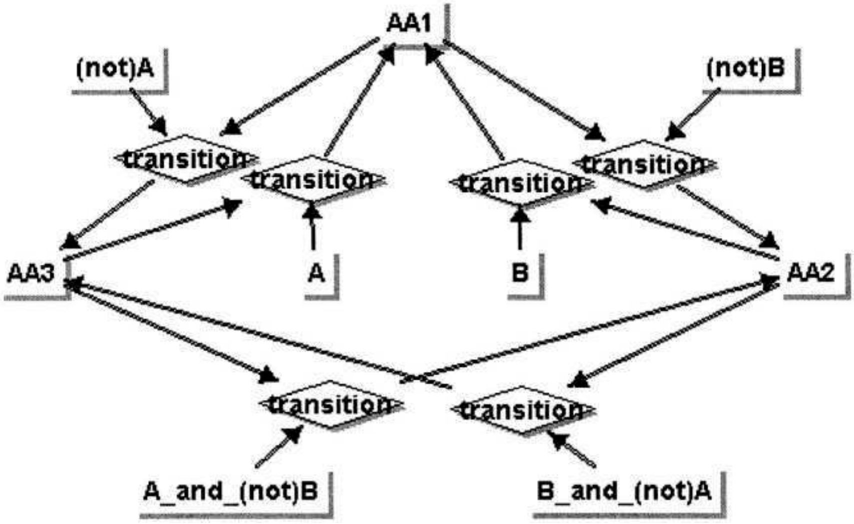


Fig. 2. CG Representation of KD45 Frame with Action-Attitudes as Possible Worlds

The possible worlds AA1, AA2 and AA3 in Figure 2 are represented by concepts that are action-attitudes within which propositions are nested. When an FIPA agent *believes* a proposition and is willing to act on that proposition, then the agent is residing in possible world AA1 with respect to that proposition. Similarly, if the agent has no knowledge of the truth value of a proposition but is willing to act; the agent resides in possible world AA2. If the agent has a *belief* (uncertain or certain) with respect to a proposition but is unwilling to act, the agent resides in possible world AA3.

3 A Rules-Based Model of Belief in Conceptual Graph Form

As with the FIPA ACL, the KQML ACL is a communication protocol expressed in terms of an agent's mental attitudes towards theirs and other agents' knowledge. However, the KQML ACL uses the mental attitudes *belief, knowledge, desire* and *intention.* The KQML mental attitude *knowledge* functions similarly to the FIPA mental attitude *belief.* No semantic models for mental attitudes are specified in KQML, but the language used to describe speech acts restricts the way mental attitudes can be used in speech acts. [27]

Initially, the KQML had no formal specification, although this was subsequently addressed. [27] As a result, there are different implementations of KQML that cannot be assumed compatible with each other. Every KQML implementation will necessarily meet some constraints on mental attitudes as an artifact of software design and implementation, although these constraints may not be explicitly stated.

The absence of an underlying logical model for mental attitudes in KQML is not a problem for our approach. We start with the same four action attitudes listed above in Table 2. A rule set can be established to govern the mental attitude *knowledge* for the particular version of KQML ACL being represented. We represent the resulting rule sets as IF-THEN statements in conceptual graph form, as shown in Figure 3. (There are several possible conceptual graph representations for IF-THEN statements, different from that shown here.)

The action-attitudes in Figure 3 will also form a context for each proposition in the KQML agent's knowledge base. We do not assume that action-attitude AA4 is not required. The particular implementation of KQML will need to be examined first, then the rule sets structured to reflect the particular KQML implementation being represented.

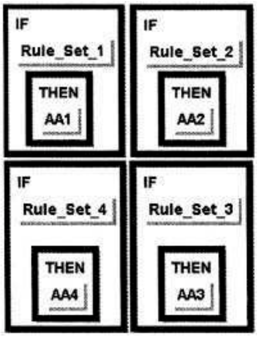


Fig. 3. Rule Sets to Define Action Attitudes

4 Knowledge Bases with Action Attitudes

The preconditions that determine when a FIPA agent may issue a communication act are stated in terms of that agent's mental attitudes towards a proposition. The proposition is located in the content field of the ACL message. [3] The knowledge base of a FIPA agent will need to correlate the conveyed propositions to the mental attitudes that the agent takes towards the propositions. The knowledge base may also correlate the mental attitudes of other communicating agents take towards these same propositions, when that information is available.

In conceptual graph form, this can be accomplished by placing propositions within the context of the agent's action-attitudes. In Figure 4, 'Allen' and 'Burt' identify two FIPA agents. A co-referent link identifies a proposition **t** that is in Allen's knowledge base and believed by Allen to also be in Burt's knowledge base. Figure 4 shows that Allen believes that he and Burt have the same action-attitude (AA2) towards proposition **t**.

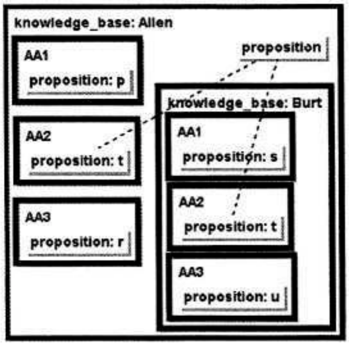


Fig. 4. Knowledge Base of Agent with Action Attitudes Providing Context

Allen's knowledge base in Figure 4 shows not only his attitudes towards propositions in his knowledge base, but also what Allen thinks are Burt's attitudes towards propositions in Burt's knowledge base. As a result, the knowledge base describes preconditions placed on FIPA communication acts, expressed in terms of mental attitudes taken by software agents towards propositions. For example, Allen may issue the FIPA ACL *inform* communication act to Burt with respect to a proposition *p*, only when Allen has action-attitude AA1 with respect to proposition *p*, and Allen does not *believe* Burt holds action-attitudes AA1 or AA3 with respect to proposition *p* (i.e., Burt has no knowledge of the truth-value of proposition *p*).

The rational effect of FIPA ACL communication acts is also expressed in terms of the agents' mental attitude towards propositions. If Allen issues the FIPA *confirm* communication act to Burt with respect to proposition *p,* a rational effect is that proposition *p* is asserted in Burt's knowledge base. Allen's model of Burt's knowledge base is not required to be accurate however; two agents will not have complete and accurate representations of each other's knowledge bases.

Although the KD45 logical model is specified for the FIPA ACL and no underlying logical model is specified for the KQML ACL, the knowledge bases of both types of agents may be represented using action-attitude contexts. The action-attitudes are established as discussed in the previous section. A step towards establishing common semantic knowledge between the two different types of agents is to determine whether the *knowledge* mental attitude of the KQML agent plays the part of the *belief* mental attitude of the FIPA agent. If these are accepted as being close, as they are in [8], then the action-attitudes of the FIPA agent may be considered to be very close to those of the KQML agent.

Since FIPA agent Allen contains within his knowledge base a representation of KQML agent Burt's knowledge base, Allen may use his representation of Burt's knowledge base to reason about communication with Burt. Allen may also evaluate whether he *believes* that common knowledge exists between Burt and himself. As an example:

If Burt has action-attitude AA2 with respect to proposition *p,* then Burt has an uncertain *belief* of proposition *p*.
If Burt has action-attitude AA2 with respect to proposition *p,* then Burt may not inform another agent of the truth-value of *p.*
Burt has action-attitude AA2 with respect to proposition **t**.
Therefore, Burt may not inform Allen of the truth-value of **t**.

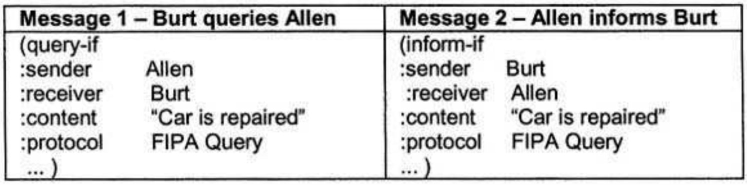
5 Two Examples of Action Attitudes in Agent Knowledge Bases

The use of action-attitude contexts to represent agent attitudes towards propositions is useful both when modeling communication between software agents that use different ACLs, and also with communication between software agents that use the same ACL. In the examples below, 'Allen', 'Burt' and 'Charlie' identify three software agents that use the FIPA ACL. The conveyed information is the sentence "Car is repaired".

5.1 Communication Between Two FIPA Agents

The sequence of messages between Allen and Burt in this first example is shown in Table 3, where the first communication act is Message 1. This message is the FIPA *query-if* communications act. Its semantic effect is to indicate that Allen is asking Burt if the statement "Car is repaired" is true. The second message is the FIPA *inform-if* communications act. Its semantic effect is to indicate that Burt is telling Allen that Burt *believes* that the statement "Car is repaired" is true.

Table 3. Two FIPA Communication acts



Message 1 – Burt queries Allen	Message 2 – Allen informs Burt
(query-if	(inform-if
:sender Allen	:sender Burt
:receiver Burt	:receiver Allen
:content "Car is repaired"	:content "Car is repaired"
:protocol FIPA Query	:protocol FIPA Query
...)	...)

Although the conveyed propositions, "Car is repaired", are identical, the two FIPA communication acts *(query-if* and *inform-if)* do not convey the same mental attitudes towards that proposition. Burt has no knowledge of the truth-value of the conveyed message; while Allen must *believe* that the statement "Car is repaired" is true. The two communication acts also have different rational effects. Once Message 2 has been issued, Allen may also 'think' that Burt *believes* that the statement "Car is repaired" is true, although Allen may not *believe* "Car is repaired" is true. In Figure 5, depicting Allen's knowledge base following Message 2, Allen is shown as *believing* the statement and also thinking that Burt *believes* the statement

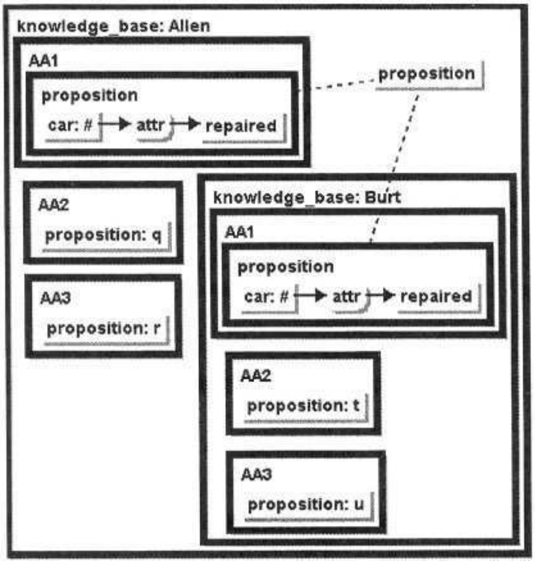


Fig. 5. Knowledge Base of Allen Following Step Two in Table Three

5.2 The FIPA Proxy Communication Act and Agent Cooperation

The FIPA describes the *proxy* communication act as having two strengths, referred to as *strong-proxy* and *weak-proxy.* Both *strong-proxy* and *weak-proxy* employ a third, middle agent to relay information between two agents. The third, middle agent is the proxy agent. *Strong-proxy* occurs when the proxy agent *believes* that the proposition it relays in the *proxy* communication act is true. *Weak-proxy* occurs when the proxy agent does not *believe* that the proposition is relays in the *proxy* communication act is true. [3] The two types of proxy communication acts are illustrated by the communication sequence shown in Table 4. The proxy agent is Burt, who is relaying information between agents Allen and Charlie.

Table 4. Weak and Strong Proxy Communications

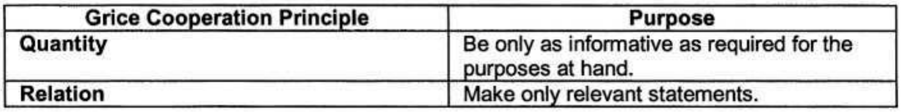
	Conversation 1: Strong Proxy	Conversation 2: Weak Proxy
1	Allen calls the repair shop to ask if his car is repaired.	(Same as Conversation 1)
2	Burt answers the phone and hears Allen's question.	(Same as Conversation 1)
3	Burt tells Charlie that Allen is asking if his car is repaired.	(Same as Conversation 1)
4	Charlie asks Burt to tell Allen that Allen's car is repaired.	(Same as Conversation 1)
5	Burt tells Allen that the car is repaired.	Burt tells Allen that Charlie says that the car is repaired.

The two conversations are identical until Step 5. This is because the decision to return a *strong* or *weak-proxy* is determined at run time. Burt is an autonomous agent who may not *believe* that Charlie is to be trusted. As a result, Burt may not *believe* Charlie's answer.

A first problem presented by Burt not *believing* Charlie is that it is a precondition on the FIPA *inform* communication act that an agent issuing an *inform*-message *believes* that the information being conveyed is true. The FIPA specification handles this problem by allowing the Burt to reply "Charlie says that the car is repaired." This is a true statement. The *inform* communication act precondition is satisfied and the *proxy* communication act may be completed as shown in Conversation 2.

A second problem is to evaluate whether Allen may detect that a *weak-proxy* has occurred. Specifically, how can Allen determine whether Burt *believes* Charlie's answer? The FIPA Communicative Act Library Specification states that the feasibility preconditions placed on communication acts have two parts; ability preconditions and context-relevance preconditions. Ability preconditions refer to the software agent's ability to execute a communication act. Context-relevance preconditions refer to the relevance of executing a communication act. The context-relevance preconditions correspond to Grice's Quantity and Relation Principles. [3] These principles, shown in Table 5, are two of four cooperation principles identified by Grice. The principles are applied to ACLs because ACL performatives are rooted in speech act theories modeled after spoken human communication. [25, 26]

Table 5. Two Gricean Cooperation Principles, referred to by the FIPA



Grice Cooperation Principle	Purpose
Quantity	Be only as informative as required for the purposes at hand.
Relation	Make only relevant statements.

Software agents that can detect when a Gricean cooperation principle has failed may conclude that a communicating agent is not cooperating. In this simple example, Allen may determine that the Quantity Principle has failed when Burt responds, "Charlie says that the car is repaired", instead of "Car is repaired." Although each statement may be true, more information is returned in the first statement than in the second. The *weak-proxy* may be detected by determining that more information is being returned than is required.

Figure 6 shows that following Step 5 of Conversation 2: (1) Allen is *uncertain* of the truth-value of the statement "Car is repaired", located in AA2; (2) Allen knows that Burt has told Allen, "Charlie says that the car is repaired." By the Gricean Quantity Principle, the additional context for the conveyed statement may indicate that either Burt or Charlie is not cooperating. As a result, (3) Allen may choose to *believe* that Burt also holds AA2 with respect to the statement "Car is repaired."

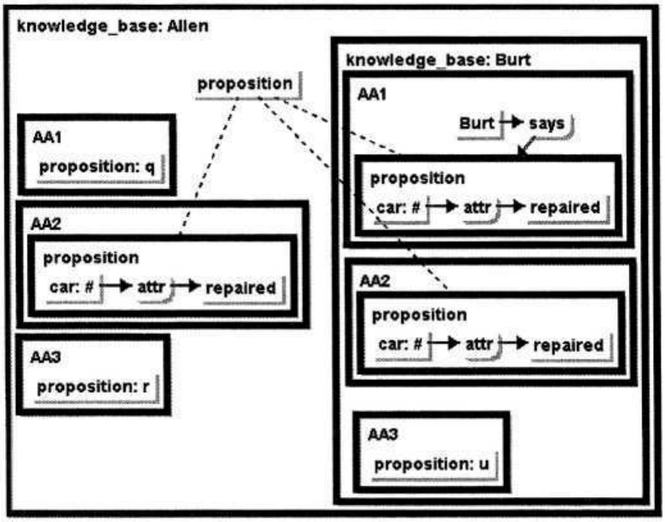


Fig. 6. Allen's Knowledge Base, Following Step Five of Conversation Two

6 Discussion

Many different formal representations have been used to model software agent constituents, at all stages of their development and implementation. For example, predicate logic has been used to represent facts about 'things' in the agent world, Petri nets or AUML have been used to represent processes and interaction protocols, OWL has been developed to represent object types and ontologies of languages [28], and programming languages such as JAVA have been used to support agent execution across a variety of computing platforms. No one representation form is sufficient for all purposes, and no unifying semantic framework for software agent technologies has been established. [29]

The efforts to establish common semantic frameworks among different types of software agents described at the beginning of this paper indicate that this is a significant goal. If software agents are to reason about themselves, other software agents and their operating environments, these agents will need access to the same types of information that designers use to reasoning about agents and their operating environments.

Although no single formalism is sufficient for every purpose, a form that may integrate several kinds of knowledge may better support agents in reasoning about their and other agent behaviors. This paper explores how representations in conceptual graph form may capture the semantics of modal *belief* logics and dynamic communication processes for software agents.

7 Conclusion

We have shown that communication between intelligent software agents is a complex process that involves semantic constituents in addition to shared vocabularies, as well as a common syntax. In our examples, we have shown the agents communicating to express both factual knowledge and mental attitudes expressed in terms of modal logics. Their knowledge bases developed in our examples show that these types of information may be represented in conceptual graph form, utilizing the demon relation to represent controlled transitions between states in a Kripke KD45 possible world *belief* model, and using concepts as contexts to represent attitudes towards propositions. In doing so, this paper addresses the larger problem identified by [23] and other researchers; which suggests that for intelligent software agents to reason about their own and other agent behaviors, the various types of knowledge used by software developers in designing and implementing software agents may also need to be provided to software agents themselves.

Acknowledgements. The authors would like to thank Lisa Cox and the anonymous reviewers for their comments and suggestions concerning this paper. Their evaluations have substantially improved the clarity ofthis presentation.

References

1. Wooldridge, M. and N.R. Jennings, *Intelligent Agents: Theory and Practice.* Knowledge Engineering Review, 1995.**10**(2): p. 115-152.
2. Finin, T., et al., *Specification ofthe KQML Agent-Communication Language.* 1993.
3. *FIPA Communicative Act Library Specification.* http://www.fipa.org/repository/fipa2000.html, 2000.
4. Genesereth, M. and S.P. Ketchpel, *Software Agents.* Communications of the ACM, 1994. **37**(7).
5. W3C, *World Wide Web Consortium.* http://www.w3.org/ , 2004.
6. Luck, M., R. Ashri, and M. D'Inverno, *Agent-Based Software Development.* 2004, Norwood, MA.: Artech House, Inc. 208.
7. Gmytrasiewicz, P., M. Summers, and D. Gopal. *Toward Automated Evolution ofAgent Communication Languages.* in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences.* 2002.
8. Reed, C.A., T.J. Norman, and N.R. Jennings, *Negotiating the Semantics of Agent Communication Languages,* in *Computatonal Intelligence.* 2002. p. 229-252.
9. Sycara, K., et al., *LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace,* in *Autonomous Agents and Multi-Agent Systems.* 2002. p. 173-203.
10. Hodgson, J.P.E., *Knowledge Representation and Language in AI.* 1991, Chichester, West Sussex, England: Simon & Simon International Group. 230.
11. Delugach, H.S., *Specifying Multiple-Viewed Software Requirements With Conceptual Graphs.* Journal of Systems and Software, 1992. **19:** p. 207-224.
12. Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine.* 1984, Reading, Mass.: Addison-Wesley. 481.
13. Sowa, J.F., *Laws, Facts, and Contexts: Foundationsfor Multimodal Reasoning,* http://www.jfsowa.com/pubs/laws.htm .
14. Dixon, C., M. Fisher, and A. Bolotov, *Resolution in a Logic of Rational Agency.* Proceedings of the 14th European Conference on Artificial Intelligence. 2000, Berlin.
15. van der Hoek, W. and M. Wooldridge, *Towards a Logic ofRational Agency.* Logic Journal of the IGPL, 2003.**11**(2): p. 135-159.

16. Cohen, P.R. and H.J. Levesque, *Intention is Choice with Commitment.* Artificial Intelligence, 1990. **42**: p. 213-261.
17. Moore, R.C., *A Formal Theory of Knowledge and Action,* in *Readings in Planning,* J.F. Allen, J. Hendler, and A. Tate, Editors. 1990, Morgan-Kaufmann: San Mateo, CA, USA. p. 473-484.
18. Rao, A.S. and M.P. Georgeff, *Modeling Rational Agents Within a BDI-Architecture,* in *Proceedings of Knowledge Representation and Reasoning,* R. Fikes and E. Sandewall, Editors. 1991, Morgan-Kaufmann. p. 473-484.
19. van Linder, B., W. van der Hoek, and J.C. Meyer, *Formalising Motivational Attitudes of Agents: On Preferences, Goals and Committments,* in *Intelligent Agents II,* M. Wooldridge and J. Muller, Editors. 1996, Springer-Verlag.
20. Bacchus, F., *Probabilistic Belief Logics,* in *European Conference on Artificial Intelligence.* 1990. p. 59-64.
21. Huth, M.R.A. and M.D. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems.* 2001, New York: Cambridge University Press. 387.
22. Wooldridge, M., *An Introduction to MultiAgent Systems.* 2002, West Sussex: John Wiley & Sons, Ltd. 348.
23. Shen, W., D.H. Norrie, and J.-P.A. Barthes, *Multi-agent Systemsfor Concurrent Intelligent Design and Manufacturing.* 2001, New York: Taylor Francis Inc. 381.
24. Franklin, S. and A. Graesser, *Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents,* in *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* 1996, Springer-Verlag: Springer-Verlag. p. 21-35.
25. Searle, J.R., *Speech acts: An Essay in the Philosophy of language.* 1969, Cambridge University Press: Cambridge.
26. Austin, J.L., *How to do things with words.* 1962, Oxford University Press: London.
27. Labrou, Y., *Standardizing Agent Communications,* in *Lecture Notes in Computer Science.* 2001. p. 74.
28. Willmott, S., I. Constantinescu, and M. Calisti. *Multilingual Agents: Ontologies, Languages and Abstractions.* in *proceedings First International Workshop on Ontologies in Agent Systems.* 2001. Montreal Canada.
29. Willmott, S., J. Dale, and P. Charlton, *Agent Communication Semantics for Open Environments.* 2002, Agentcities.RTD Project; http://www.agentcities.org/EURTD/Pubs/eurtd.02.willmott.semantics.pdf.

Improving Generalization Level in UML Models
Iterative Cross Generalization in Practice

Michel Dao[1], Marianne Huchard[2], M. Rouane Hacène[3], C. Roume[2], and
Petro Valtchev[3]

[1] France Télécom R&D, DAC/OAT, 38-40 av. Général Leclerc,
92794 Issy-les-Moulineaux cedex 9, France
michel.dao@francetelecom.com
[2] LIRMM, UMR 5506,161 rue Ada, 34392 Montpellier cedex 5, France
{huchard,roume}@lirmm.fr
[3] DIRO, Université de Montréal, C.P. 6128, Succ. "Centre-Ville", Montréal, Canada, H3C 3J7
{rouanehm,valtchev}@iro.umontreal.ca

Abstract. FCA has been successfully applied to software engineering tasks such as source code analysis and class hierarchy re-organization. Most notably, FCA puts mathematics behind the mechanism of abstracting from a set of concrete software artifacts. A key limitation of current FCA-based methods is the lack of support for relational information (e.g., associations between classes of a hierarchy): the focus is exclusively on artifact properties whereas inter-artifact relationships may encode crucial information. Consequently, feeding-in relations into the abstraction process may substantially improve its precision and thus open the access to qualitatively new generalizations. In this paper, we elaborate on ICG, an FCA-based methodology for extracting generic parts out of software models that are described as UML class diagrams. The components of ICG are located within the wider map of an FCA framework for relational data. A few experimental results drawn from an industrial project are also reflected on.

1 Introduction

Current trends in object-oriented software construction, namely MDA (Model-Driven Architecture)-based approaches, promote designing high-level models that represent domain and application concepts ("Platform Independent Models"). These models, typically described in UML (Unified Modeling Language), are further on mapped to the target implementation platform ("Platform Specific Models"). Modeling has thus become a key activity within the software process whereas large efforts are currently spent in developing automated tools to assist it.

Formal Concept Analysis (FCA) has already been successfully applied to the analysis [1] and restructuring [2,3,4,5,6,7] of conceptual class models: it helps reach optimal hierarchical organization of the initial classes by discovering relevant new abstractions. However, providing far-reaching abstraction mechanisms requires the whole feature set of UML to be covered, inclusive those encoding relational information (e.g., UML associations), whereas such features clearly outgrow the scope of standard FCA.

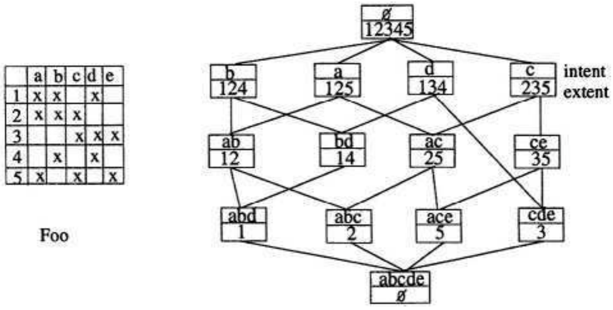
K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 346–360, 2004.

Making FCA work on UML models is the global aim of our study. Here, we propose a new relationally-aware abstraction technique, ICG *(Iterative Cross Generalization),* which works on several mutually related formal contexts that jointly encode a UML class diagram. It performs simultaneous analysis tasks on the set of contexts where inter-context links are used to propagate knowledge about the abstractions from a context into its related contexts (and thus broaden the discovery horizon on those contexts).

The paper recalls the basics of FCA (Section 2) before providing a motivating example (Section 3). Our recent FCA-based framework for processing relational data is presented in Section 4. In Section 5 we specify ICG while emphasizing the role UML meta-model plays in data description within ICG. Experiments done in the framework of industrial projects are then reported (Section 6) with a discussion of benefits and difficulties in applying ICG.

2 FCA and Class Hierarchy Restructuring

Formal concept analysis (FCA) [8] studies the way conceptual structures emerge out of observations. Basic FCA considers an *incidence* relation I over a pair of sets O (*objects,* further denoted by numbers) and A (*attributes,* denoted by lower-case letters). Binary relations are introduced as formal contexts $\mathcal{K} = (O, A, I)$. An example of a context, Foo, is provided in Figure 1 on the left, where a cross in i-th line / j-th column means that the object i has the attribute j.



	a	b	c	d	e
1	x	x		x	
2	x	x	x		
3			x	x	x
4		x		x	
5	x		x		x

Fig. 1. A sample context and the Hasse diagram of its concept lattice

A pair of derivation operators, both denoted by $'$, map sets of elements between A and O by performing intersections on the corresponding sets of rows/columns. For instance, $\{1,2\}' = \{a,b\}$ and $\{a,c\}' = \{2,5\}$. The $'$ operators define a *Galois connection* [9] between 2^A and 2^O, whereby the component operators $''$ satisfy the closure properties. The underlying sub-families of closed sets are bijectively mapped to each other by $'$ with pairs of mutually corresponding closed sets termed *(formal) concepts.* More precisely, a concept is a pair *(X, Y)* from $2^O \times 2^A$ with $X = Y'$ and $X' = Y$, where X is the *extent* and Y is the *intent.* The set $\mathcal{C}_{\mathcal{K}}$ of all concepts from $\mathcal{K}$ is partially ordered by the inclusion of extents, while the resulting ordered structure $\mathcal{L}_{\mathcal{K}} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ is a complete lattice with joins and meets based on intersection of concept intents and extents, respectively. The lattice of the Foo context is drawn in Figure 1 on the right (as a Hasse diagram).

Research on applications of FCA has yielded a set of meaningful substructures of the concept lattice. For instance, in object-oriented software engineering, the assignments of specifications/code to classes within a class hierarchy is easily modeled through a context, and applying FCA to a particular hierarchy may reveal crucial flaws in factorization [2] and therefore in maintainability. The dedicated substructure that specifies a maximally factorized class hierarchy of minimal size is called the Galois sub-hierarchy (GSH) of the corresponding context. Mathematically speaking, the GSH is made out of all the extremal concepts that contain an object/attribute in their extents/intents: $\{(o'', o') | o \in O\} \cup \{(a', a'') | a \in A\}$.

Moreover, as practical applications of FCA may involve processing of non-binary data, many-valued contexts have been introduced in FCA. In a many-valued context $\mathcal{K} = (O, A, V, J)$, each object o is described by a set of attribute - value pairs (a, v), meaning that J is a ternary relation that binds the objects from O, the attributes from A and the values from V. The construction of a lattice on top of a many-valued context requires a pre-processing step, called scaling, which basically amounts to encoding each non-binary attribute by a set of binary ones.

3 Improving UML Models: A Motivating Example

A highly simplified example introduces the problem domain. Consider the UML model in Figure 2. A class Diary is associated to a class Date through the association orderedBy. Class Date has three attributes (or variables) day, month and year and two methods including isLeapYear() and a comparison method <(Date). Another class Clock is linked to Time class via the association shows. Class Time is described by the three attributes hour, min and sec, and by a method <(Time) which aims at comparing times.

Current approaches for applying formal context analysis to this UML model would lead to the formal context ofFigure 3: classes are the formal objects while UML attributes, methods and association ends are the formal attributes (names have been reduced to their first letters). This formal context does not reveal any new concept, although comparison methods < indicate that a *magnitude* concept is underlying the model and that diaries and clocks are devices which manipulate magnitudes.

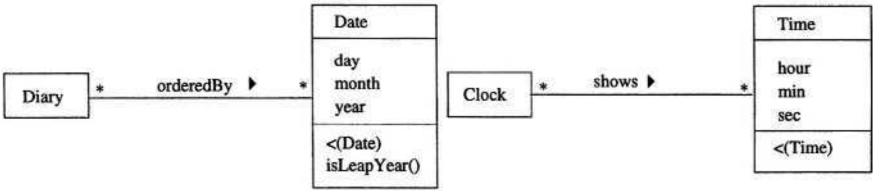
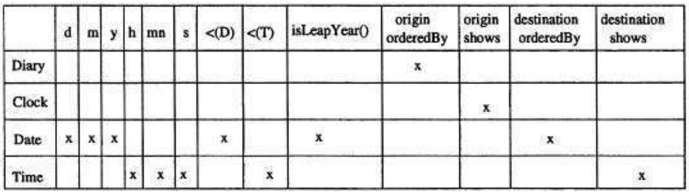


Fig. 2. Diary and clock



	d	m	y	h	mn	s	<(D)	<(T)	isLeapYear()	origin orderedBy	origin shows	destination orderedBy	destination shows
Diary										x			
Clock											x		
Date	x	x	x				x		x			x	
Time				x	x	x		x					x

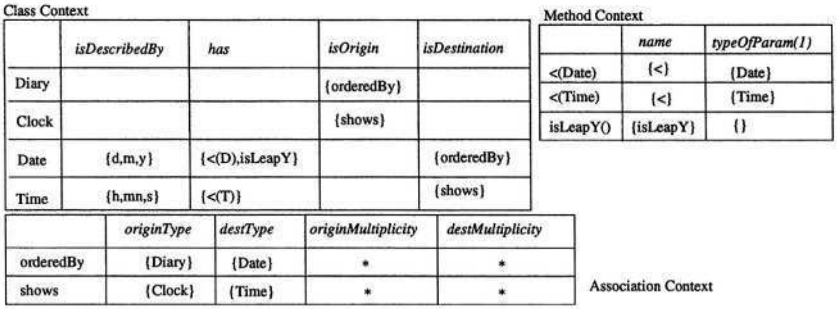
Fig. 3. Formal context for diary and clock

To infer a more elaborate UML model, we apply an approach that may be summarized as follows. On the one hand, we process various sorts of UML entities such as attributes, methods and associations, as first-class formal objects and assign a formal context to each entity sort. Moreover, we use *relational attributes* to express links between entities and model them as inter-context binary relations.

On the other hand, we use a repeated scaling along the relational attributes to propagate the knowledge about possible generalizations between related contexts. Thus, the concept construction process amounts to alternating scaling and proper construction until stability in concept structures is reached.

In Figure 4, three many-valued formal contexts describe classes, associations and methods as first-level formal objects, respectively. Here, UML class attributes are not processed as objects for simplicity sake, but in the general case they are. Note that some formal attributes (*e.g.* `originType`) are relational ones while others are not (*e.g.* `originMultiplicity` or `name`). Figure 5 shows the main relational attributes of the example.

Scaling techniques are used to transform many-valued contexts into binary ones. Values of each many-valued attribute are represented as the objects of scale context where the formal attributes are important properties of these values. In Figure 6, values of `typeOfParam(1)` are scaled considering the specialization order (inheritance) on classes: a value (a class *C*) for `typeOfParam(1)` is associated in the scale with all super-classes of *C* and *C* (`All` represents the top of the class hierarchy). Note that the available class organization is replicated in the scale lattice (which basically represents a



Class Context

	isDescribedBy	*has*	*isOrigin*	*isDestination*
Diary			{orderedBy}	
Clock			{shows}	
Date	{d,m,y}	{<(D),isLeapY}		{orderedBy}
Time	{h,mn,s}	{<(T)}		{shows}

Method Context

	name	*typeOfParam(1)*
<(Date)	{<}	{Date}
<(Time)	{<}	{Time}
isLeapY()	{isLeapY}	{}

	originType	*destType*	*originMultiplicity*	*destMultiplicity*
orderedBy	{Diary}	{Date}	*	*
shows	{Clock}	{Time}	*	*

Association Context

Fig. 4. Formal contexts for classes, methods and associations

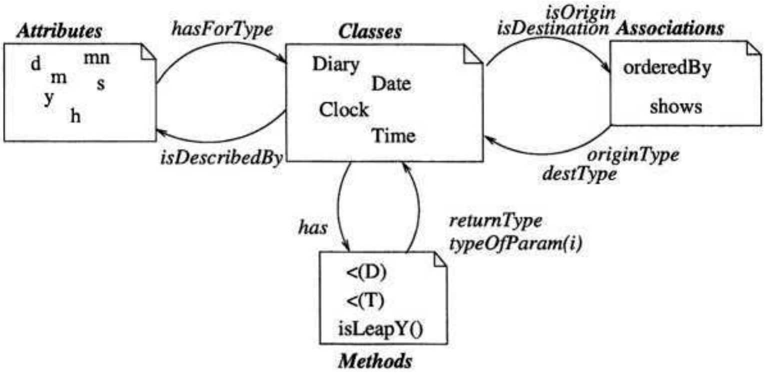


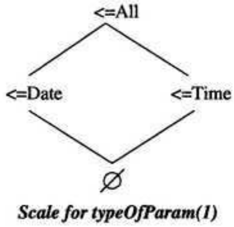
Fig. 5. Relations between the formal contexts

	name	*typeOfParam(1)*
<(Date)	{<}	{Date}
<(Time)	{<}	{Time}

Many–valued Method Context

	<=Date	<=Time	<=All
Date	x		x
Time		x	x

Scale Context for typeOfParam(1)



Scale for typeOfParam(1)

	name:<	*typeOfParam(1):Date*	*typeOfParam(1):Time*	*typeOfParam(1):All*
<(Date)	x	x		x
<(Time)	x		x	x

Scaled Method Context

Fig. 6. Scaling the method context

nominal scale). The concept lattice of the scaled method context (see Figure 7) contains four concepts: `m4` represents `isLeapYear()`; `m1` and `m2` represent both initial comparison methods, respectively, and `m3` introduces a generalized method <. Bottom and top are skipped since useless here.

The method lattice (Figure 7) is now used as a scale for the formal attribute `has` owned by classes. Thus, if a class has a method *meth* in the initial many-valued context, then it owns all the formal attributes `has` :`m` in the scaled class context where `m` stands for a method concept whose extent contains the formal object representing *meth*. The resulting scaled class context and its lattice (with top and bottom dropped) are shown in Figure 8. The lattice includes a new concept `c1` which obviously represents comparable objects, hence it could be called `Magnitude`.

Our knowledge about the concept structure on classes has thus grown and the new abstractions can be used as descriptors that could, whenever shared, induce potential abstractions on related contexts. For example, the method context could be fed with the

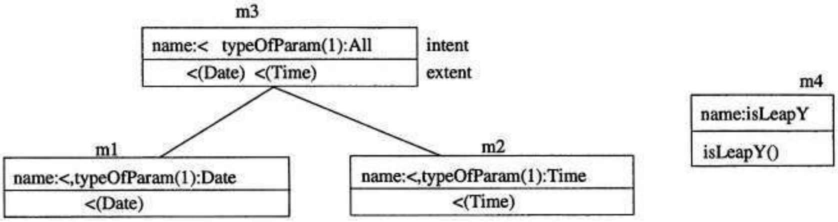

m3
name:< typeOfParam(1):All
<(Date) <(Time)
intent
extent
m4
name:isLeapY
isLeapY()
m1
name:<,typeOfParam(1):Date
<(Date)
m2
name:<,typeOfParam(1):Time
<(Time)

Fig. 7. The first concept lattice on methods

	d	m	y	h	mn	s	*has:* m1	m2	m3	m4	*isOrigin:* orderedBy	shows	*isDestination:* orderedBy	shows
Diary											x			
Clock												x		
Date	x	x	x				x		x	x			x	
Time				x	x	x		x	x					x

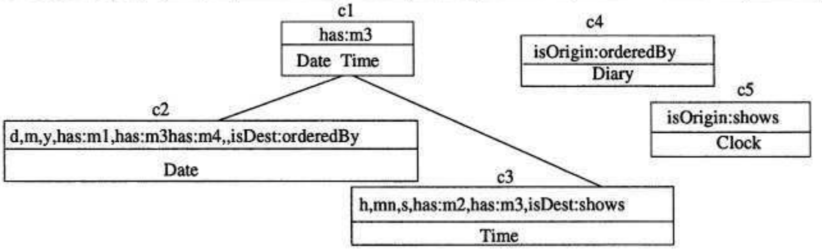

c1
has:m3
Date Time
c4
isOrigin:orderedBy
Diary
c5
isOrigin:shows
Clock
c2
d,m,y,has:m1,has:m3has:m4,,isDest:orderedBy
Date
c3
h,mn,s,has:m2,has:m3,isDest:shows
Time

Fig. 8. Scaling the class context (up), class concept lattice (down) without the top and the bottom

knowledge about Magnitude thus prompting a re-consideration of its conceptual structure. Thus, three new binary attributes typeOfParam(1):c2, typeOfParam(1):c3 and typeOfParam(1):c1 replace the initial ones (typeOfParam(1):All, typeOfParam(1):Date and typeOfParam(1):Time). The resulting concept lattice remains isomorphic to that of Figure 7, however its concepts are explicitly related to existing concepts on classes, e.g., the top concept intent is bound to c1 via typeOfParam(1):c1.

The same procedure can be applied for scaling the association context, revealing that the two formal objects can be generalized by a new association which ends into the c1 concept. The scaling of isOrigin and isDestination from the class context, using the augmented association lattice, introduces a new generalization of Diary and Clock (representing devices which manipulate magnitudes). The resulting set of abstractions, re-interpreted in UML, is shown in Figure 9. The associations orderedBy and shows are linked to the new association manipulate by the constraint subset which indicates a specialization relationship. Because of this constraint, their names are now prefixed by the symbol"/", used in UML for highlighting elements that derive from others.

To sum up, we may claim that the apparent commonalties between the classes `Time` and `Date` have led to the constitution of common superclass, `Magnitude`. The discovery of this class has been propagated to both method and association contexts where new abstractions have been created to reflect the existence of `Magnitude`. Finally, these new abstractions reflected reversely on classes where a superclass of `Diary` and `Clock` emerged. New generalizations are especially useful in design: *e.g.* new general classes given with their abstract methods can serve as type for writing generic code; new general associations clarify the model as specializations like `orderedBy` and `shows` can disappear in overviews of the model; new classes can factorize attributes, methods, associations added in further development, etc.

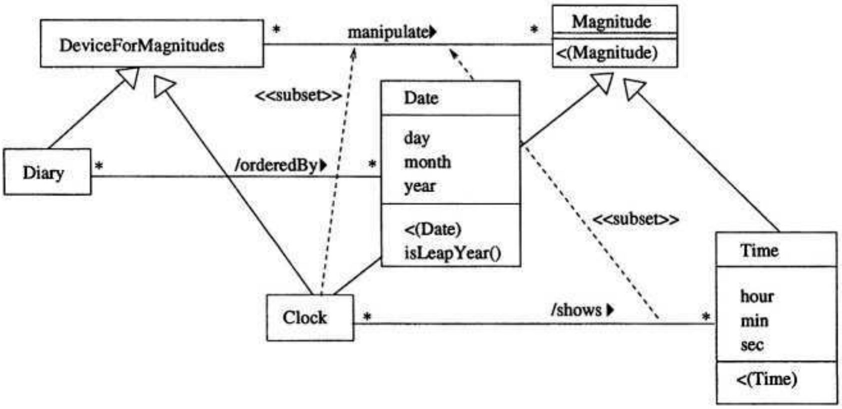


Fig. 9. Diary and clock after iterative cross generalization

4 Bringing Relational Concepts to Core FCA

In the following, we summarize the key elements of our relational FCA framework. A detailed description could be found in [10].

As in classical FCA, heterogeneous datasets, i.e., ones made out of several sorts of individuals, are introduced through a family of contexts, one per sort of formal objects. Here, a set of binary relations (or set-valued functions) is added to data description, which map objects from a context to sets of objects from another one.

Definition 1 (Relational context family).
A relational contextfamily $\mathcal{R}^s$ is a pair $(\mathbf{K}_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}})$ where $\mathbf{K}_{\mathcal{R}}$ is a set of s multi-valuedcontexts $\mathcal{K}_i = (O_i, A_i, V_i, J_i)$ and $\mathcal{A}_{\mathcal{R}}$ is a set ofp relationalattributes (set-valuedfunctions) α_j such thatfor each j, $1 \leq j \leq p$ thereexist r and q in $[1, s]$ with $\alpha_j : O_r \to 2^{O_q}$.

For instance, the data in our running example (see Section 3) constitute a RCF with four contexts and nine relations.

Conceptual scaling [11] is a FCA technique that transforms a many-valued context $\mathcal{K} = (O, A, V, J)$ into binary one $\mathcal{K}^d = (O, A^d, I^d)$ by replacing non-binary attributes from A by a set of binary ones, called *scale attributes.* Scale attributes basically describe meaningful features of the values of the initial attribute, say a, and therefore induce a lattice of concepts, called the *scale lattice,* on top of the value set $V(a)$. By bringing those attributes to the objects from O, conceptual scaling allows new concepts to occur in which the members of the extent share abstractions of the initial values rather than values themselves. Clearly, the choice of scale attributes has a direct impact on the structure of the target concept lattice: different attribute sets may lead to different lattices.

The same principle may be applied to the processing of relations which are basically object-valued attributes: given a relation $\alpha : O_r \rightarrow 2^{O_q}$ and $o \in O_r$ the set $\alpha(o)$ could be replaced by a collection of binary attributes that characterize it. As the entire process is ultimately aimed at detecting commonalties in the abstractions that conceptually describe the target objects, the scaling binds scale attributes to existing concepts on the co-domain context rather than to formal attributes of this context (see the attributes `typeOfParam(1):cX` from Section 3). Moreover, as we argued in [10], the most natural choice for the scale lattice of α is the lattice of the context $\mathcal{K}_q$ since it embeds the most precise knowledge about the meaningful abstractions on the set O_q. However, in specific situations smaller structures, such as the GSH, may be more appropriate.

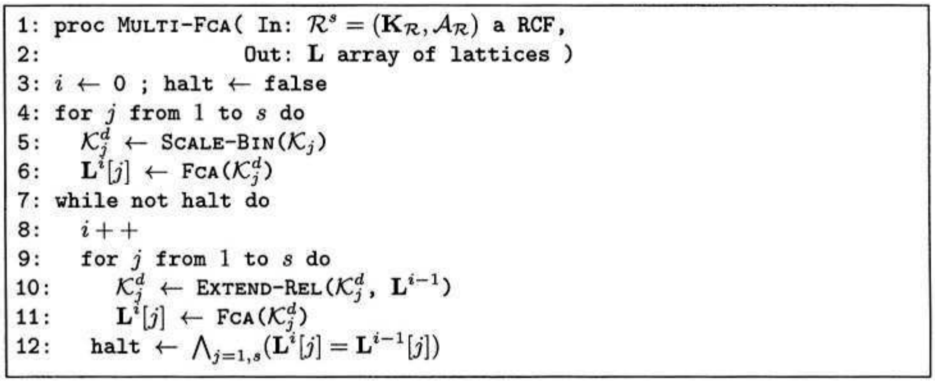
Consider an object o_i from $\mathcal{K}_r$ and its encoding in terms of concepts from $\mathcal{K}_q$.Thus, given a concept c_j from $\mathcal{L}_q$, the corresponding binary attribute $\langle \alpha : c_j \rangle$ will be incident to o_i depending on the way the image set of objects $\alpha(o_i)$ compares to $Extent(c_j)$. Two different encoding schemes are possible, i.e., o_i gets $\langle \alpha : c_j \rangle$ whenever: 1) $\alpha(o_i) \subseteq Extent(c_j)$ ("narrow") or 2) $\alpha(o_i) \cap Extent(c_j) \neq \emptyset$ ("wide"). The "narrow" scheme clearly fits lattice-shaped scales whereas the "wide" one suits also less extensive concept structures.

To sum up, the encoding by concepts rather than by formal attributes from the destination context $\mathcal{K}_q$ eases the interpretation of the formal concepts discovered in the source context $\mathcal{K}_r$. Moreover, such an encoding fits a step-wise discovery of the scale concepts as illustrated in Section 3: the formation of some new concepts within $\mathcal{K}_q$, e.g., through refining of the descriptions of the context objects, results in the addition of new scale attributes in the encoding of O_r along α.

Given a relational context family $\mathcal{R}^s$, our aim will be to construct s lattices of formal concepts $\mathcal{L}_i$, $(0 \leq i \leq s)$, one per context $\mathcal{K}_i$, such that the concepts reflect both shared attributes and similarities in object relations, i.e., common concepts in the co-domain context. Obviously the relational scaling helps to reduce the lattice construction on relational data to the binary case so that the same algorithmic procedures could be applied. However, unlike conventional constructions, some RCF may require a step-wise construction process due to the mutual dependencies between contexts, as it was shown in the UML model analysis. Indeed, having aligned scales with actual concept hierarchies on the destination contexts, an apparent deadlock occurs whenever two contexts are connected both ways by a pair of relational attributes (or chains of such attributes). For instance, in Figure 5, the class context is doubly connected to the method one by the initial attribute pair (`typeOfParam(i)`,`has`).

To resolve the deadlocks resulting from circularity in the relational structure of a RCF, we apply a classical fixed-point computation mechanism that proceeds stepwise. Its grounding principle lies in the gradual incorporation of new knowledge gained through scaling: the computation starts with uniformly nominal scales for all relational attributes and at each subsequent step uses the previously discovered concept structures within the respective co-domain contexts as new and richer scales.

Technically speaking, the global lattice extraction process associated with a RCF alternates between relational scaling and lattice construction (see Algorithm 1). At the initial step, relations are ignored (line 5), hence the lattices at this stage (line 6) are not impacted by the relational information and rather reflect common non-relational attributes. At the following step these lattices are used as new scales for a first-class relational scaling thus providing new possibilities for generalizations (lines 10-11). The scaling (line 10) / construction (line 11) steps go on until the global set of concepts stabilizes, i.e., for each context $\mathcal{K}_j$ the lattice $\mathbf{L}^i[j]$ at the i**-th** step is isomorphic to the one of the $i - 1$**-th,** $\mathbf{L}^{i-1}[j]$ where $\mathbf{L}^i$ denotes the array of contexts at step i. Stabilization of the process can be deduced from the fact that the formal objects do not change over the steps; the concept number of the lattice associated with $\mathcal{K} = (O, A, V, J)$ is bounded by $2^{min(|O|,|A \times V|)}$, which gives a bound to the scaling of relational attributes.



```
1: proc MULTI-FCA( In: $\mathcal{R}^s = (\mathbf{K}_\mathcal{R}, \mathcal{A}_\mathcal{R})$ a RCF,
2:                 Out: $\mathbf{L}$ array of lattices )
3: $i \leftarrow 0$ ; halt $\leftarrow$ false
4: for $j$ from 1 to $s$ do
5:   $\mathcal{K}_j^d \leftarrow$ SCALE-BIN($\mathcal{K}_j$)
6:   $\mathbf{L}^i[j] \leftarrow$ FCA($\mathcal{K}_j^d$)
7: while not halt do
8:   $i++$
9:   for $j$ from 1 to $s$ do
10:     $\mathcal{K}_j^d \leftarrow$ EXTEND-REL($\mathcal{K}_j^d$, $\mathbf{L}^{i-1}$)
11:     $\mathbf{L}^i[j] \leftarrow$ FCA($\mathcal{K}_j^d$)
12:   halt $\leftarrow \bigwedge_{j=1,s}(\mathbf{L}^i[j] = \mathbf{L}^{i-1}[j])$
```

Algorithm 1: Construcion of the set of concept lattices corresponding to a RCF.

Once the lattices of all contexts are available, a post-processing step clarifies the links between concepts induced by relational scale attributes. In fact, many concept intents will present redundancies: a concept c from $\mathcal{K}_r$ related, via $\langle \alpha : c_1 \rangle$, to c_1 from $\mathcal{K}_q$ will necessarily be related to all the super-concepts of c_1. Thus, for each super-concept c_2 of c_1, c will possess the attribute $\langle \alpha : c_2 \rangle$. As the latter reference does not add new information with respect to $\langle \alpha : c_1 \rangle$ to c_1, it may be deleted. This means that in the intent of c, among all the binary attributes $\langle \alpha : c_i \rangle$, only those corresponding to minimal

c_i will be preserved. For instance, in Figure 8, the attribute `has :m3` is redundant in the intent of class concept `c2` since `c2` also owns `has :m1`, whereas `m3` is a super-concept of `m1` in the method lattice.

5 Specifying the Iterative Cross Generalization Process

UML class diagrams (models) in more details. In class diagrams, classes are associated to structural features (attributes) and behavioral features (operations and methods). In Figure 10 (top) the main elements of attribute and method description are presented: visibility (+, - and #); attribute types *e.g.* `String`, `Point` or `Color` which can be classes; return type; parameter type list; multiplicity for many-valued attributes (like `color`), the multiplicity is a set of integer intervals restricting the value number (for color, multiplicity **`1..*`** expresses the fact that color has one or more value); static status (underlined feature); derived status (introduced by /).

Figure 10 (bottom) also illustrates the main aspects of UML associations. An association is composed of at least two association ends. When it has a name (for example `place_order`), the name is followed by a triangle which establishes the direction for reading this name: a person places an order and not the other way round. An association end is typically characterized by: a type (the class *C* involved in this end), for example `Person` and `Order` are the two end types of the association `place_order`; a visibility; a multiplicity; a navigability (shown through an arrow next to the type end); a white or black diamond which indicates an aggregation or a composition. An association end is sometimes provided with a role name which gives more accurate semantics to objects when they are involved in the link, *e.g.* role *employee* for a person in association *manage*. When the association owns variables and methods it is considered as an association class, *e.g.* `Access` is an association class that supports the variable `passwd`.

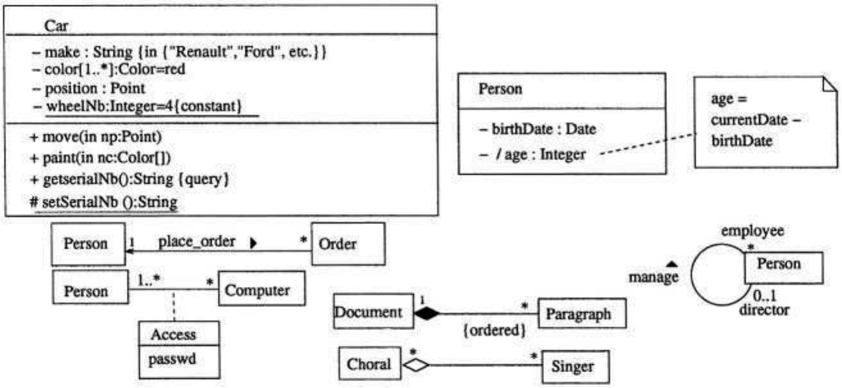


Fig. 10. Classes and associations

Using the UML meta-model to guide the context construction. The definition of UML is established by the UML meta-model, that is a model that defines the language for models. The UML meta-model is described through a subset of UML, and is given with well-formedness rules in the formal language OCL (Object Constraint Language), as well as with semantics in natural language. Part of this meta-model [12] relevant to our problem, that considers the classes and their features, is shown in Figure 11. The meta-class `Class` specializes `Classifier`, and as such, inherits from the possibility to own `Features` (`Attribute` or `Method`). An `Attribute` includes the meta-attributes `initialValue`, `multiplicity`, `visibility`, `changeable`; it has a type via the meta-association that links it to `Classifier`. A `Method` has the meta-attributes `body`, `isQuery`, `visibility`, and is composed of an ordered set of `Parameters`. An `Association` is composed of several `AssociationEnds` which have a type which is a classifier. `AssociationEnds` are described by a type (a classifier), and several meta-attributes including `isNavigable`, `isOrdered`, `aggregation` and `multiplicity`.

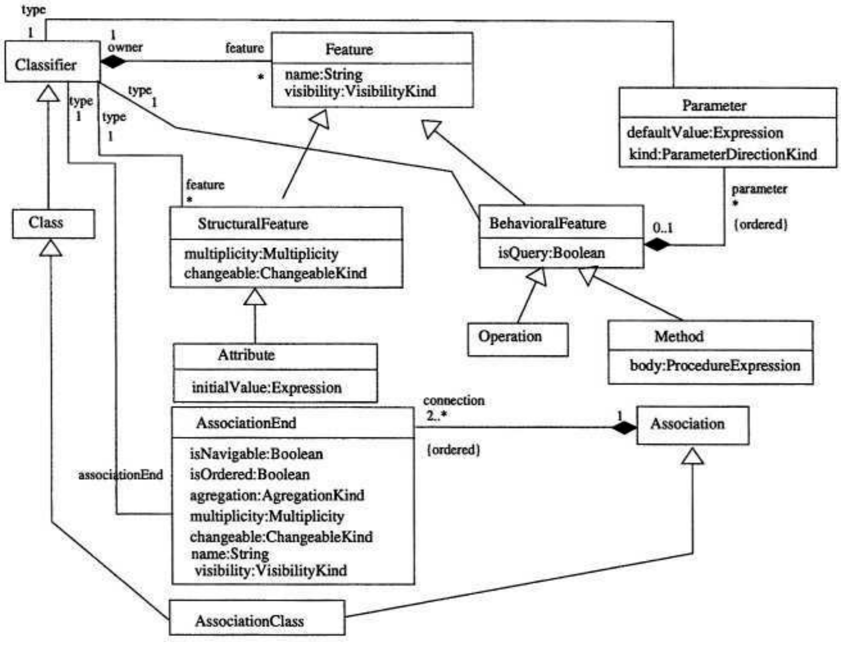


Fig. 11. Extracts from the UML meta-model

As a meta-description of UML, the meta-model naturally contains the good abstractions for determining the right formal contexts: meta-classes are straightly interpreted as formal objects, while meta-attributes and ends of meta-associations are their formal attributes. Nevertheless, such an approach can lead to the manipulation of many tables of data, and to the use of descriptors that generate too numerous uninteresting concepts. `Parameter` for example is preferably included in the description of methods. Associations should be described by an ordered set of association ends, but if we consider only binary and directed associations, as often suggested in modeling [13], we can avoid having a specific formal context for association end description. Conversely, if we want to inspect all possible generalizations of associations in the general case, a formal context describing association ends would be relevant.

In the context of the MACAO project[1], we have considered the relational context family $(\mathcal{K}_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}})$ defined as follows. $\mathcal{K}_{\mathcal{R}} = \{\mathcal{K}_C, \mathcal{K}_{At}, \mathcal{K}_M, \mathcal{K}_{As}\}$. $\mathcal{K}_C = (O_C, A_C, V_C, J_C)$ is the formal context on classes; A_C is empty in our current experiments. $\mathcal{K}_{At} = (O_{At}, A_{At}, V_{At}, J_{At})$ corresponds to the formal context on attributes. A_{At} includes the formal attributes *name, multiplicity, initialValue* corresponding to the UML meta-attributes. V_{At} contains the possible values for these formal attributes, *i.e.* possible attribute names, unions of integer intervals, etc. $\mathcal{K}_M = (O_M, A_M, V_M, J_M)$ describes methods. A_M includes the formal attributes *name, body,* while V_M contains possible method names, and expressions that represent method bodies. $\mathcal{K}_{As} = (O_{As}, A_{As}, V_{As}, J_{As})$ is the formal context on associations. As we have chosen in our first experiments to consider binary directed associations, the two association ends are called *origin* and *destination* and their description is integrated into the formal context for associations. Formal attributes are then *name, nameOrigin, isNavOrigin, isOrderedOrigin, multOrigin,* and symmetrical attributes for destination end. $\mathcal{A}_{\mathcal{R}}$ is the set of relational attributes that relate the previous contexts. They are found using the meta-associations between meta-classes `Classifier` and `Attribute`, `Classifier` and `Association`, or `Classifier` and `Method` (going through `Parameter`). They have been presented on the edges in Figure 5. For example we have *has* : $O_C \rightarrow 2^{O_M}$ or *originType* : $O_{As} \rightarrow 2^{O_C}$.

6 Experiments

The ICG procedure has been implemented in the Java-based **Galicia**[2] platform [14] and connected to the UML CASE tool Objecteering as part of the MACAO project, thus enabling application of ICG to class diagrams designed within Objecteering. Thus, for a given UML class diagram, RCF is exported[3] in a format which is readable by ICG, which is run and its results are imported back in Objecteering in order to create a new class diagram which can then be studied and compared to the original one. We present here some results of the application of ICG on several medium sized projects of France Télécom. Three different projects have been used for these experiments [15]: *project 1* deals with the management of an information system, ICG was applied to the design model of this project (the model used for Java code generation); *project 2* concerns an intranet software that was also in its design stage; *project 3* is a prospective project regarding the elaboration of a common user data model for several telecommunication

[1] A joint project of France Télécom, SOFTEAM and LIRMM supported by the French department of research and industry (RNTL); http://www.lirmm.fr/˜macao.

[2] See the web site at: `http: //www. iro.umontreal. ca/~galicia.`

[3] A limited configuration of RCF is possible within Objecteering.

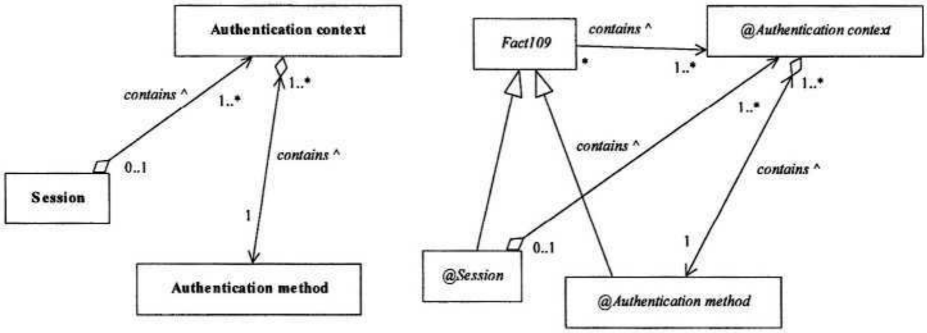


Fig. 12. Factorization of an association

services. We have applied ICG to several class hierarchies of project 3: four class hierarchies of service-specific models and the class hierarchy of the common model being specified.

The results of the ICG implementation were shown to the designers of the class hierarchies who gave an appreciation regarding the relevance of the proposed restructuring with respect to the semantics of the underlying data model. The class hierarchies of those projects consist of a few dozens of classes and the number of new UML elements created by ICG (attributes, methods, classes, inheritance links) may vary from a few to several hundreds in some cases, involving a tedious work of selection and interpretation.

Several new factorization classes or associations proposed by ICG were found absolutely relevant by the class diagram designers. For instance, Figure 12 shows the factorization of an association. Left part shows the initial state of the few classes involved and the right part shows the proposed restructuring. The ICG algorithm properly proposes to factorize both associations named `contains^`through the creation of a new class `Fact109` connected to the class `@Authentication context` through a new association with the same name. Notice that the algorithm may propose to factorize role names depending on the way the designer has named the associations: association names, role names or both. This corresponds to the formal attributes *name, nameOrigin* and *nameDestination* of the formal context on associations. The multiplicity on the side of the class `@Authentication context` is also properly factorized into a **`1..*`** multiplicity. On the other hand, one may question the factorization of `0..1` and `1` multiplicities into * (it could have been factorized into `0..1`) but this is an internal choice of the algorithm that could be fine-tuned.

7 Conclusion

We presented a new FCA-based technique (ICG) which processes several mutually related formal contexts and sketched its application to UML class diagram restructuring. Experiments on industrial-scale projects established the feasibility of our approach (execution time and semantic relevance of the results) and highlighted the crucial role of parameter tuning and appropriate user interface. A key track of improvement is the separation of formal attributes that guide the construction of new abstractions *(e.g.* names, types of attributes, association ends, etc.) from secondary ones that only help to increase the precision (*e.g.,* multiplicity or navigability). Another current concern is the integration of a domain ontology into the ICG framework that should enable the comparison of symbolic names used by the designer. This is crucial for any automated reconstruction technique such as our, because terms are not uniformly used over UML diagrams, many synonymy, homonymy or polysemy situations occur. Although Objecteering offers an operational user interface for ICG there is a large space for improvement. First, designers that are FCA neophytes would benefit from an automated assistance in tool fine-tuning. Second, navigation and edition tools should help make the entire ICG process more interactive and thence more purposeful, e.g., by supporting run-time filtering of the discovered abstractions.

References

1. Snelting, G., Tip, F.: Understanding class hierarchies using concept analysis. ACM Transactions on Programming Languages and Systems **22** (2000) 540–582
2. Godin, R., Mili, H.: Building and maintaining analysis-level class hierarchies using Galois lattices. In: Proceedings of OOPSLA'93, Washington (DC), USA. (1993) 394–410
3. Dicky, H., Dony, C., Huchard, M., Libourel, T.: On Automatic Class Insertion with Overloading. In: Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'96. (1996) 251–267
4. Godin, R., Mili, H., Mineau, G., Missaoui, R., Arfi, A., Chau, T.: Design of Class Hierarchies Based on Concept (Galois) Lattices. Theory and Practice of Object Systems **4** (1998)
5. Huchard, M., Leblanc, H.: Computing Interfaces in Java. In: Proc. IEEE International conference on Automated Software Engineering (ASE'2000), 11-15 September, Grenoble, France. (2000) 317–320
6. Yahia, A., Lakhal, L., Cicchetti, R., Bordat, J.: iO2 - An Algorithmic Method for Building Inheritance Graphs in Object Database Design. In: Proceedings of the 15th International Conference on Conceptual Modeling ER'96. Volume 1157. (1996) 422–437
7. Yahia, A., Lakhal, L., Bordat, J.: Designing Class Hierarchies of Object Database Schemas. In: 13 ièmes journées Bases de Données Avancées. (1997) 371–390
8. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1999)
9. Barbut, M., Monjardet, B.: Ordre et Classification: Algèbre et Combinatoire. Volume 2. Hachette (1970)
10. Valtchev, P., Rouane, M.H., Huchard, M., Roume, C.: Extracting Formal Concepts out of Relational Data. In SanJuan, E., Berry, A., Sigayret, A., Napoli, A., eds.: Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR), 3-6 September, INRIA (2003) 37–49
11. Ganter, B., Wille, R.: Conceptual Scaling. In: Applications of combinatorics and graph theory to the biological and social sciences. Volume 17 of The IMA volumes in Mathematics and its applications., New York (1989) 139–167
12. Rational Software Corporation: UML v 1.3, Semantics. version 1.3 edn. (1999)
13. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object Oriented Modeling and Design. Prentice Hall (1991)

14. Valtchev, P., Grosser, D.,Roume,C.,Hacene, M.R.: GALICIA: an open platform for lattices. In B. Ganter, A.d.M., ed.: Using Conceptual Structures: Contributions to 11th Intl. Conference on Conceptual Structures (ICCS'03), Aachen (DE), Shaker Verlag (2003) 241–254
15. Dao, M.: Validation sur de grands projets, Projet MACAO (RNTL). Technical Report sous-projet MACAO 5.1, France Télécom R&D (2003)

KNAML: A Knowledge Representation Language for Distributed Reasoning

Gordon Streeter and Andrew Potter

Sentar, 4900 University Square, Suite 8,
Huntsville, Alabama 35816 USA
{gstreeter, apotter}@Sentar.com
http://www.sentar.com/

Abstract. The Knowledge Agent Mediation Language (KNAML) is designed for use in multi-agent reasoning systems. Like conceptual graphs, KNAML represents knowledge using concepts, relations, and graphs. Concepts and relations are linked to form graphs, and graphs may be nested within other graphs. Additional constructs are used to support distributed reasoning and ontological concision. KNAML treats ontologies as knowledge domains that happen to be of the ontology domain. It uses an ontology of ontologies to define the concept and relation types available in an ontology. KNAML knowledge resources are modular to facilitate rapid development and efficient inter-agent processing. KNAML supports ontological specification of an extensible set of knowledge modalities, such as workflows, decision trees, and graphs that reflect the processing specializations of various knowledge agents and supports multi-modal knowledge authoring. Implemented in Java, KNAML supports subsumption, unification, and binding operations required by the host multi-agent system to carry out knowledge discovery and synthesis.

1 Introduction

Multi-agent systems that perform distributed reasoning pose distinct challenges for knowledge representation languages. Because the agents share a distributed knowledge corpus, the language must support explicit ontologies. To facilitate agent diversity, these ontologies should define, in addition to domain conceptualizations, various structural features of the language as well. Because contributions to multi-agent systems may come from a variety of organizations, the resources used to implement the language should use widely accepted technologies. Reliance on esoteric programming languages should be avoided. Even so, the knowledge representation language must readily support advanced reasoning capabilities, such as subsumption, unification, and binding. As with monolithic knowledge systems, the language should also support human readability and understandability. Finally, the knowledge representation language should support agent interoperability through the use of standard network transmission and data storage.

The Knowledge Agent Mediation Language (KNAML) is designed specifically for use in multi-agent reasoning systems. As with conceptual graphs [1], KNAML represents knowledge using concepts, relations, and graphs. Concepts and relations

K.E. Wolff et al. (Eds.): ICCS 2004, LNA1 3127, pp. 361–374, 2004.

may be linked together to form graphs, and graphs may be nested within other graphs or concepts. Additional constructs, including concept frames and arc labels, are provided to support distributed reasoning and ontological concision. Frames are used to implement relational instantiation, which enables the system to provide discrete handling for factual assertions. Arc labels are syntactic helpers used to facilitate non-ambiguous mapping between ontological structures and knowledge content.

KNAML supports knowledge capture and agent specialization by implementing an extensible set of knowledge modalities, such as workflows, rules, decision trees, and graphs. This is accomplished by using an ontological specification for each modality. Each modality is accompanied by a corresponding editor that enables the user to create integrated knowledge projects, consisting of a set of multi-modal knowledge modules defined to address an anticipated range of problems using a multi-agent architecture. At the storage and transmission level, KNAML is represented as XML. KNAML is being used in a variety of applications now under development. Editors for workflows and graphs are now in use, and additional editors are in development.

This paper describes KNAML and its use in the KnoWeb® multi-agent architecture. The KnoWeb architecture uses mediated reasoning to integrate a variety of agent capabilities. These include knowledge bases, databases, UML workflows, and sensors. The Java implementation of KNAML supports subsumption, unification, and binding. The result is a simple but highly expressive language for representing knowledge for performing automated reasoning in a distributed environment.

2 Background: A Multi-agent Architecture

KnoWeb is a multi-agent system architecture that uses mediated reasoning to perform dynamic decision-making [2]. KnoWeb employs a small group of core agents to implement its reasoning model, and engages a loosely coupled confederation of specialist agents to carry out goal-driven and event-driven tasks. The core agents consist of a Meta Agent, a service agent, and one or more domain advisors. The *Meta Agent* provides domain-neutral mediation and conflict resolution. In its role as mediator, the Meta Agent enlists other agents to satisfy goals presented by a requesting agent. It takes care of inter-agent coordination and planning needed to reach a goal. By concentrating reusable intelligence in this central resource, redundant complexity in specialist agents is reduced. The *service agent* maintains a registry of agent capabilities used to provide brokerage services. The service agent is functionally similar to matchmaking agents described in [3], however, in KnoWeb agents typically are both information providers and requesters, resulting in agent interaction that may be intensively cooperative. The *domain advisor* provides conflict resolution and planning strategies used by the Meta Agent. Agents communicate using the Knowledge Agent Mediation Language (KNAML) developed by Sentar.

KnoWeb mediated reasoning is implemented in Java. The Java implementation of KNAML supports subsumption, unification, and binding. Subsumption uses existential conjunctive logic to establish the truth value of one graph based on the known value of another. Graph and sub-graph unification supports discovery by indicating how one graph is subsumed by another. Binding is used for joining graphs to produce knowledge synthesis. The binding operation also supports backtracking. This is necessary to assure that graphs are recoverable in the event of binding failure.

The reasoning process used by the Meta Agent is straightforward. Throughout the process, the Meta Agent uses an agenda to keep track of what it is doing and why, and it maintains a context of asserted propositions. The process consists of several phases: initiation, alliance, marshalling, resolution, and response. The process begins when an agent initiates a request. The agent does so by sending the request to the Meta Agent. When the Meta Agent accepts a request, it first checks to see if the answer is already in the context or if the request is already in its agenda. If the answer is already in the context, the Meta Agent uses it to generate a response. Otherwise, it proceeds with problem solving.

In the alliance phase, the Meta Agent selects a relevant domain advisor for use in planning and conflict resolution. This alliance is sometimes necessary because the Meta Agent is domain neutral, and both planning and conflict resolution involve domain specific considerations. Typically, the domain agent is an expert system constructed specifically for a problem domain.

Following the alliance phase, the Meta Agent performs marshalling. Marshalling consists in identifying the agents to enlist in handling the request. Registered capability and ontological consistency are among the criteria. Also as part of marshalling, the Meta Agent negotiates with the domain advisor to determine whether any agents should be excluded from request processing. This affords the domain advisor an opportunity to eliminate extraneous branches from the agenda before it is executed by the Meta Agent. An agent's registered capabilities may include preconditions that must be resolved as part of request processing. These preconditions may require the Meta Agent to spawn additional requests, so the Meta Agent must be able to maintain recursive contexts.

Upon completion of marshalling, the Meta Agent dispatches the request to the enlisted agents. These agents, if they choose to handle the request, attempt to instantiate it, and return their results to the Meta Agent. The agents may initiate nested requests as needed, and the Meta Agent will invoke the reasoning process for each of these requests.

In the resolve phase, the Meta Agent discards duplicate responses and passes the remaining responses to the domain advisor for evaluation. The domain advisor may discard additional responses. The domain advisor may also initiate further nested requests which must be serviced prior to resolution of the original request. The remaining responses are asserted into the context and the solution is dispatched to the agent that initiated the request.

Note that elsewhere in the literature the term "Meta Agent" is used to refer to an agent that reasons about other agents [4] or as an agent that aggregates other agents [5]. Although this latter definition might have some functional applicability here, to the extent that multiple Meta Agents could operate among intersecting or complementary agent clusters, no other architectural support for this concept seems necessary. And for reasoning about other agents, no particular kind of agent is required. What would be required are appropriate knowledge resources and some stock of problems to consider.

3 Knowledge Agent Mediation Language

KNAML is based on conceptual graphs, as defined by Sowa [1]. In KNAML, knowledge is represented using concepts, relations, and graphs. Concepts and relations may be linked together to form graphs, and graphs may be nested within other graphs. Ontological support is built into the language. The result is a simple but expressive language for the representation of complex knowledge.

A concept may represent any entity. Concepts have a type and a referent. The type is the ontological category to which instances of the concept belong. The referent denotes a specific instance or set of instances of a concept. The following graph contains a single concept. The concept type is Person and the referent is #Bob.

```
[
    [Person:#Bob]
]
```

In the above example, the type of concept referent used is called an indexical. Indexicals are always preceded by the "#"sign, e.g. #Bob. Indexicals may be used to designate individual concepts. KNAML supports two other kinds of referents. These are string literals and descriptors. String literals are represented using character strings enclosed in quotes. Descriptors are graphs. These are used frequently in KNAML. In the following example, the referent of the concept type Proposition is a descriptor. As shown, descriptors can be nested:

```
[
    [Proposition:[
        (Believes)
            +-believer-->[Person:#Jack]
            +-belief-->[Proposition:[
                (GoingTo)
                    +-traveler-->[Person:#Bob]
                    +-destination-->[City:#Boston]
            ]]
    ]]
]
```

Frames are implemented using relations, with arcs for each slot. Frames are used for relational instantiation. This makes it possible for the system to distinguish one otherwise identical instance of a relation from another. If there are multiple assertions, for example, that "Bob is going to Boston," possibly received from differing agents or from the same agent at different times, relational instantiation permits the system to uniquely identify each trip. Further, frames provide a convenient way to specify properties for each instance, such as time and date or mode of transportation. So, if the ontology for concept type Person specifies associations with concepts of type Address, City, State, and Zip, the association can be defined like this:

```
[Person: #Bob [
     (PersonalDetails)
          +-name-->[Name: "Robert McNamara"]
          +-address-->[Address: "1000 Defense Pentagon"]
          +-city-->[City: "Washington"]
          +-state-->[State: "DC"]
          +-zip-->[Zip: "20301"]
]]
```

Relations are used to define relationships among concepts. Each relation has a type and a collection of arcs. The arcs are used to define the linkage between a relation and its concepts. Arcs are labeled. The labels are determined in the ontological definition of the relation. Each arc terminates in a concept. The types of each of these concepts are also specified ontologically.

Here, the relation `GoingTo` has two arcs, one labeled `traveler` and the other `destination`. The `traveler` arc terminates on a concept of the Type `Person`, with referent of `#Bob`. The destination arc terminates on a concept of type destination, with a referent of # `Boston`.

```
[Proposition:[
     (GoingTo)
          +-traveler-->[Person:#Bob]
          +-destination-->[City:#Boston]
]]
```

Suppose that several people are going to Boston. One way to express this would be to create several graphs, one for each traveler. Another would be to define the ontology to permit the use of a plural. A plural is a special concept in which all the elements are of a specified type:

```
[
   [Proposition: [
       (GoingTo)
            +-travelers-->[Person: { #Bob, #Carol, #Ted,
                                     #Alice }]
               +-place-->[City: #Boston]
   ]]
]
```

A graph is a container for concepts, relations, and other graphs. A graph is represented by a matching set of square braces. Here is a graph containing a relation and its ontologically specified set of concepts:

```
[

     (GoingTo)
         +-traveler-->[Person:#Tex]
         +-destination-->[City:#Madison]
]
```

4 Ontology

As previously suggested, ontology is central to KNAML. Use of an explicit ontology enforces consistency within a knowledge module, and more importantly for multi-agent systems, ontology makes it possible for agents to share knowledge. An ontology is a specification of the concepts comprising a domain and the interrelationships that may hold between these concepts. We treat an ontology as a knowledge module whose domain happens to be an ontology. To author such an ontology, an ontology-ontology is required.

An Ontology consists of an ontology name and a collection of concept and relation types. The name is a concept of type TypeName, and the concept and relation types are of ConceptType and RelationType respectively:

```
[Ontology:[
      (OntologyFrame)
            +-aName-->[TypeName:]
            +-someConceptTypes-->[ConceptType:{*}]
            +-someRelationTypes-->[RelationType:{*}]
]]
```

A concept type is a concept of type ConceptType. To define a concept type, the ConceptType concept type is used. The ConceptType has two parts, a type name and a type. The name is a concept of type TypeName, and the type is ontologically unspecified and could be anything. By convention, the type must be a relation type, an enumeration, a primitive, or nothing at all.

```
[ConceptType:[
      (ConceptTypeFrame)
            +-aName-->[TypeName:]
            +-aType-->[]
]]
```

A relation type is a concept of type RelationType. A relation type specification consists of a name and a collection of arc types. The name is a concept of type TypeName. The arcs are concepts of type ArcType.

```
[RelationType:[
      (RelationTypeFrame)
            +-aName-->[TypeName:]
            +-someArcTypes-->[ArcType:{*}]
]]
```

An arc type is a concept of type ArcType. The specification of an arc type consists of a label, an arc terminus, and a plural indicator. The label is a concept of type TypeName. The terminus is a concept of type ConceptType. The plural indicator is a concept of type Boolean.

```
[ArcType:[
     (ArcTypeFrame)
          +-aLabel-->[TypeName:]
          +-aTerminus-->[ConceptType:]
          +-aPluralIndicator-->[Boolean:]
]]
```

There are several primitive types used to provide the ontological foundation. These include `TypeName`, `Boolean`, and `Enumeration`. A concept of type `TypeName` is a string, a concept of type `Boolean` may be either true or false, and a concept of type `Enumeration` may be used to define a specific set of values that may be applied to a concept.

Using the ontology-ontology, it becomes possible to treat an ontology as a knowledge module. This means new ontologies can be authored using the same tools used to create other knowledge modules, and further, it is possible for a knowledge agent to reason about ontologies just as they do about other domains. We anticipate being able to apply this approach to ontology reusability problems.

5 Knowledge Modalities

In a multi-agent system, agent specializations may occur along lines of knowledge domains. For example, one agent might specialize in some area of product diagnostics, and another could specialize in customer service. The two agents combined would be useful in creating a product support application. However, there is another form of specialization, one which occurs along lines of knowledge modalities. That is to say, the agents specialize in their forms of knowledge representation. Some problems are best solved by using rules, others by using decision trees, and still others respond well to workflows. The possibilities are unlimited. A single form of knowledge representation, no matter how powerful, is insufficient. Using the wrong representation leads to poor design, difficult knowledge authoring, and poor system performance.

KNAML supports an extensible set of modalities, such as workflows, rules, decision trees, and graphs. This is accomplished by using KNAML ontological support to create ontologies for specialized modalities. Each modality is accompanied by a corresponding editor that enables the user to create integrated knowledge projects, consisting of a set of multi-modal knowledge modules defined to address a predefined range of problems using a multi-agent architecture.

For example, the workflow modality allows knowledge to be authored, expressed, and processed as workflows, using workflow symbology. The workflow agent implements UML activity diagrams, including actions, forks, merges, branches, joins, and transitions. Workflow activities and transition guards include goals, which are expressed as KNAML graphs. At runtime, these goals are evaluated—using the Meta Agent reasoning process where appropriate—and the results are used to determine the path taken by the workflow. That multi-agent systems would benefit from a well-defined agent interaction protocol is clear [6]. The workflow agent orchestrates the behavior of the multi-agent system, and it does so in an architecturally neutral manner. A workflow is a tactical plan for solving a problem. By specifying the steps required to solve the problem, the order in which they are to be taken, and the conditions under which they will be invoked, the workflow provides a coherent approach to agent cooperation. Because all activities performed by other agents (possibly including other workflow agents) are mediated through the Meta Agent, workflows can maintain goal-level visibility into the problem solving process. This simplifies the knowledge representations required by individual agents and reduces the need for extensive preconditions on agent capabilities. Supporting the workflow agent is a workflow editor used to create workflow modules.

Thus, in addition to support knowledge processing, the multi-modal approach makes knowledge representations more intuitive for non-logicians. We believe this is a significant benefit, especially in contrast to knowledge representations which rely exclusively on description logics, markup languages, or some combination of the two, as is the case with the Resource Description Framework, as described in [7] and elsewhere.

6 Reasoning in KNAML

The Java implementation of KNAML supports subsumption, unification, and binding. Subsumption uses existential conjunctive logic to establish the truth value of one graph based on the known value of another. Graph and sub-graph unification supports discovery by indicating how one graph is subsumed by another. Binding is used for joining one graph with another. The binding operation also supports backtracking.

We have chosen to implement KNAML using Java. This commitment is consistent with our general ground-rule that our development be portable, practical and accessible. Implementing unification in Java has required that subsumption, unification, and binding be addressed explicitly, whereas, in a logic programming language such as Prolog, these capabilities might have been left to fend for themselves. Here we discuss some of the issues associated with defining and implementing the KNAML reasoning capability.

6.1 Subsumption

Subsumption is used for graph comparison. For example, it may be used to compare a possible solution to a goal. If the goal can subsume the possible solution, then the possible solution is, in fact, a solution. This technique has been used in the Meta Agent to check a new goal against the context, to see if the solution is already known. Subsumption is a very specific test for similarity in two conceptual structures. A conceptual structure p is said to subsume the structure q if the following conditions hold:

1. If p and q are concepts, then p subsumes q if the type of p subsumes the type of q and the referent of p subsumes the referent of q.

a. The type of p subsumes the type of q if the former is unspecified or if the two are identical.

b. The referent of p subsumes the referent of q if any of the following are true:

i. The referent of p is unspecified.

ii. The referents of p and q are identical primitives.

iii. The referents of p and q are subsumable plurals.[1]

iv. The referents of p and q are subsumable graphs.

2. If p and q are relations, then p subsumes q if the type of p is identical to the type of q and the arcs of p subsume the arcs of q. The arcs of p subsume the arcs of q if all of the following are true:

   a. The number of arcs in p is equal to the number of arcs is q.

   b. There is a one-to-one match from the arc labels in p to the arc labels in q.

   c. The concept at the end of each arc with a given label in p subsumes the concept at the end of the arc with the same label in q.

3. If p and q are graphs, then p subsumes q if one of the following conditions hold:

   a. The content of p is unspecified.

   b. Both p and q are empty.

   c. p and q are non-empty, and for every structure in p, there is a corresponding structure in q which the structure in p subsumes.

6.2 Unification

Unification supports discovery by producing a new graph which shows how one graph is subsumed by another. In the Prolog programming language, both unification and subsumption lend themselves to backtracking. For example, the matching of graphs is not order dependent, so an attempt may begin in one order until it fails, then another order is attempted. However, there is no backtracking in Java—once an object is changed, it is changed. Therefore the Java implementation of unification does not change the arguments being unified, but produces a third, unified argument. If at any point during the unification of p and q to produce r, some substructures p' and q' are being unified in an attempt to produce r' and the attempt succeeds, r' can be added to r. But if it fails, any structures accumulated into r' can be discarded, and any other appropriate attempt can be made.

Interestingly, when p is unified with q to produce r, it would seem that r would be identical to q. We might draw this conclusion from a cursory reading of the rules for subsumption, which say that p is subsumes q if p is identical to q, or, in some

[1] The definition for subsumption of plurals is not given here, but it is essentially the same as subsumption of graphs.

specific instances, if p is unspecified. In the instances in which p and q are identical, r will have the same value as p and q, and so, obviously, it will have the same value as q. In the instances where p is unspecified, q may be specified or unspecified, but in either case, r will have the value of q. So, in all cases, r will have the same type and structure as q. What unification allows r to inherit from p is indexicals.

Consider the case of an agent that can convert from degrees Fahrenheit to degrees centigrade. To make the agent capability declaration simple, let us say that the agent can convert both directions, and also can check a given pair of numbers to see if they are paired by the conversion process (that is, that the conversion of one would result in the other). The first of the following three propositional functions represents this capability, the second represents a goal for the conversion of 32 degrees Fahrenheit to centigrade, and the final propositional function represents the unification of the two:

```
[
     [PropositionalFunction:#p[
          (Convert)
               +-degreeF-->[#theFarValue]
               +-degreeC-->[#theCentValue]
     ]]
     [PropositionalFunction:#q[
          (Convert)
               +-degreeF-->["32"]
               +-degreeC-->[]
     ]]
     [PropositionalFunction:#r[
          (Convert)
               +-degreeF-->[#theFarValue"32"]
               +-degreeC-->[#theCentValue]
     ]]
]
```

In keeping with the earlier discussion, these are labeled “#p”, “#q”, and “#r”, respectively. Notice that unification takes the values from q, the goal, which is supplied by the system, but retains the indexicals from p, the capability, which was supplied by the agent. This allows the agent to use the indexicals to quickly locate important concepts in the goal, even though unification may have a different form from the original capability. Granted, it would not be difficult to locate any concept in this example. Actual capabilities are generally not quite so simple.

6.2.1 Unspecified Graphs

A unique requirement that has emerged in our use of KNAML is the need to be able to work with unspecified graphs. The KNAML code has for some time been aware of “null” graphs, graphs which have not yet had their element vector set. In various places, the code attempts to treat these graphs the same as empty graphs. The concept of unspecified graphs is borrowed from plurals, where there is both the empty plural “{}” and the unspecified plural **“{*}”**. Sowa [1, 8] does not include this notion in his definition of graphs, and is silent on the whole idea of propositional functions. Further, if graphs can be made to be a true superset of functionality to plurals (and why should they not?) then plurals would be redundant, and could be replaced by sequences. Sequences are similar to plurals, but are ordered. These could be important, for example, when we wish to send a list of options to the user, and we wish those options to be presented to the user in the same order they were sent.

6.2.2 Unification by Sub-graph

Unification by sub-graph is important as a vehicle for discovery and a test of truth. What is true for a graph should also be true for a sub-graph. For example, consider a goal `p` formed from the question, "Who is going to Boston and Chicago?" The proposition `q`, presumably from context, shows that it is known that Bob is going to Baltimore, Boston, Chicago, and Washington:

```
[PropositionalFunction:#p [
     (Going)
         +-traveler-->[#theTraveler]
         +-destination-->[[
             ["Boston"]
             ["Chicago"]
         ]]
 ]]
 [Proposition:#q [
     (Going)
         +-traveler-->["Bob"]
         +-destination-->[[
             ["Baltimore"]
             ["Boston"]
             ["Chicago"]
             ["Washington"]
         ]]
 ]]
```

Obviously, if Bob is going to Baltimore, Boston, Chicago and Washington, then Bob is going to Boston and Chicago, and `p` should unify with `q`, as shown here in `r`:

```
 [Proposition:#r [
     (Going)
         +-traveler-->[#theTraveler"Bob"]
         +-destination-->[[
             ["Baltimore"]
             ["Boston"]
             ["Chicago"]
             ["Washington"]
         ]]
 ]]
```

6.3 Binding

Because unification does not affect p or q, it is very useful and relatively simple to implement. However, there are times when its usefulness is limited for this very reason. We have found we need an operation where p is “bound” to q, where the process of binding is like unification, except that the result, rather than being directed to a new structure r, is directed back into p. An example of this is found in the behavior of the workflow agent.

Consider a section of a workflow that monitors the temperature of some piece of equipment. As a failsafe, this section of the workflow takes two different temperature readings, one in Fahrenheit and one in centigrade, and compares the two to see if they agree. Presumably, if they do not agree, the workflow would generate some alarm, but we will concern ourselves only with the states in the workflow which take the two readings and compare them. Here are the goals associated with the states of interest:

```
[
     [PropositionalFunction:#goal1 [
          [Sensor:#theSensor[
               (Property)
                    +-aValue-->[#degreesF]
                    +-aName-->["DegreesFahrenheit"]
                    +-aConcept-->[#theSensor]
          ]]
     ]]
     [PropositionalFunction:#goal2 [
          [Sensor:#theSensor[
               (Property)
                    +-aValue-->[#degreesC]
                    +-aName-->["DegreesCentigrade"]
                    +-aConcept-->[#theSensor]
          ]]
     ]]
     [PropositionalFunction:#goal3 [
          (Convert)
               +-degreesF-->[#degreesF]
               +-degreesC-->[#degreesC]
     ]]
   ]
```

Note that the goals are joined. That is, the third goal shares concepts with the first two. When the third goal is executed, it needs the values returned from the execution of the first two goals. The workflow agent is an abstract agent, and knows nothing about any particular domain, so it certainly will not know that it needs to obtain these values for this particular case. The agent could be implemented to unify the goal with the result, then search future goals for indexicals found in the unified result, copying the value from the unified result to those goals, but this would be a messy process.

If, instead, the agent would, after execution of each goal, bind the goal with the result, future goals which are joined to the goal would automatically be changed. For example, if the Fahrenheit reading were 32, after the execution of the first goal and the binding of the goal with the result, the concept with the indexical "degreesF" would be bound to the value "32". Similarly, if the reading for degrees centigrade were "0", after the execution of the second goal and the binding of the goal with the result, the concept with indexical "degreesC" would be bound to "0". This would set the third goal up to test the conversion of 32 degrees Fahrenheit to 0 degrees centigrade, which is exactly what we would want.

6.3.1 Binding and Subsumption

Since binding is destructive, it is useful to know in advance if it will succeed. Although it might seem that subsumption could be used as an indicator of success, there are problems with such an approach. Binding, like unification and subsumption, is not order dependent. The algorithm must attempt to find an order in which the graphs will bind. The subsumption and unification algorithms can simply try progressive orders until one succeeds, or until all have failed. Unlike subsumption and unification, binding changes the graph while in the process of trying progressive orders. This means that binding may not succeed, even when subsumable order has been found. Thus we have found it necessary to implement backtracking for binding.

7 Conclusion

In this paper we have described KNAML and its use in the KnoWeb multi-agent architecture. KNAML supports knowledge capture and agent specialization by implementing an extensible set of modalities, such as workflows, rules, decision trees, and graphs. This is accomplished by using an ontological specification for each modality. Each modality is accompanied by a corresponding editor that enables the user to create integrated knowledge projects, consisting of a set of multi-modal knowledge modules defined to address an anticipated range of problems using a multi-agent architecture. At the storage and transmission level, KNAML is represented as XML. KNAML is being used in a variety of KnoWeb applications now under development. Workflows and graph editors are now in use, and additional editors are in development.

The KnoWeb architecture integrates a variety of agent capabilities. These include knowledge bases, databases, UML workflows, and sensors. The Java implementation of KNAML supports subsumption, unification, and binding. Subsumption uses existential conjunctive logic to establish truth value of one graph based on the known value of another. Graph and sub-graph unification supports discovery by indicating how one graph is subsumed by another. Binding is used for joining graphs to produce knowledge synthesis. The binding operation also supports backtracking. This is necessary to assure that graphs are recoverable in the event of binding failure.

The result is a simple but highly expressive language for representing knowledge for performing automated reasoning in a distributed environment.

References

[1] J. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA: Brooks/Cole, 2000.

[2] G. Streeter, A. Potter, and T. Flores, A mediated architecture for multi-agent systems, presented at Seventeenth International Joint Conference on Artificial Intelligence: Workshop on E-Business and the Intelligent Web, Seattle, WA, 2001.

[3] K. Sycara, M. Klusch, S. Widoff, and J. Lu, Dynamic service matchmaking among agents in open information environments, ACM SIGMOD Record, vol. 28, pp. 47-53, 1999.

[4] J. Dix, V. S. Subrahmanian, and G. Pick, Meta Agent Programs, Journal of Logic Programming, vol. 46, pp. 1-60, 2001.

[5] J. Sutherland and W.-J. van den Heuvel, Enterprise application integration and complex adaptive systems, Communications of the ACM, vol. 45, pp. 59-64, 2002.

[6] G. W. Mineau, Representing and enforcing interaction protocols in multi-agent systems: An approach based on conceptual graphs, presented at IEEE/WIC International Conference on Intelligent Agent Technology, Halifax, Canada, 2003.

[7] S. Decker, Melnik, M., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdman, M. Horrocks, I., The Semantic Web: The Roles of XML and RDF, in IEEE Internet Computing, 2000, pp. 63-74.

[8] J. Sowa, Conceptual structures: Information processing in mind and machine. Reading, MA: Addison-Wesley, 1984.

Interoperability of Ontologies Using Conceptual Graph Theory

Dan Corbett

Intelligent Systems Laboratory
School of Computer and Information Science
University of South Australia
Adelaide, South Australia 5095

Abstract. One of the main goals in achieving smarter management and retrieval of information for the future is in ensuring interoperability among knowledge bases. The achievement of this ultimate goal in knowledge management requires the development of semantic representations and methods for comparison which will allow a semantic interoperability of knowledge bases. This paper will examine technologies which can be used to implement these goals. We explore current technologies that lie behind developments in subsumption theory, intelligent cooperating agents and ontologies, and explore how these technologies can be applied to the problems facing knowledge management. This paper offers a treatment of the semantics of comparison for the ontologies underlying knowledge bases. Our goal is to match and filter information retrieved from a knowledge base by using an ontology created by the user. This will require defining means to compare, link or merge ontologies. This work is significant in that our formal definition of ontology dispenses with the class/instance boundary, thus saving complexity, while allowing a filtering mechanism for facilitating interoperability.

Keywords: Knowledge representation, automated reasoning, ontology, knowledge servers

1 Introduction

With the large amount of information (and knowledge) available through the Internet, users are starting to look for effective ways to filter through the information, to find only the information relevant to their work. Instead of using the web to provide documents and raw data, users will instead use a knowledge server (or knowledge provider [1]) to tailor the knowledge retrieved to the user's specific purposes.

Many authors in the field of knowledge representation have described the use of constraints and filters to tailor information for a user. The goal of that work is to tailor each knowledge base to provide the information that the user is looking for. One way to do this would be simply to make all knowledge bases conform to a common standard. Since forcing all knowledge base authors everywhere to adopt a single, uniform standard is a near-impossible task (consider all of the current web stan-

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 375-387, 2004.

dards) it is more reasonable to attempt to create methods to allow a knowledge base to find points of interoperability with other knowledge bases automatically.

It has been stated clearly and convincingly that communications between knowledge bases is more than a matter of agreeing to a common representation standard [1, 2]. True interoperability depends on the semantic representation of the knowledge and of the domain represented in the knowledge base. Ontologies have been used to define a framework for the knowledge in a domain, but so far very little work has been done in the area of the interoperability of ontologies, or in ontology comparison or reuse [3].

This article offers a formal treatment of the semantics for the ontologies underlying knowledge bases. We define methods which allow an interoperability between knowledge bases created by different knowledge engineers. Our goal is to filter out information retrieved from a knowledge base by using an ontology created by the user, in order to retrieve only the tailored information that the user is after. This will require defining means to compare, link or merge ontologies. This work is significant in that it goes beyond previous work which is often limited to strict taxonomies, or which is constrained to particular types of inheritance or structures. We define methodologies which go beyond these restrictions.

2 Comparing and Tailoring Ontologies

Graphical representation models have been used for knowledge representation for several reasons. Mugnier and Chein [4] list the main advantages as a solid grounding when it comes to combinatorial algorithms, and that a graph (as a mathematical object) allows a natural representation. This natural representation allows not only the construction of effective algorithms over the knowledge, but also assists the user in understanding the knowledge, making it easier for the user to manipulate, edit and revise the knowledge. Graphical representations will also allow an explicit representation of the concepts in a domain, and their relations.

Graphical representations can take many forms, but are mostly based on various association mechanisms which are used to identify relations among objects and classes in a domain. Associations such as deduction and constraints can be implemented through graph matching and through the use of subsumption hierarchies.

There has been some previous work in formalizing the definitions and operations on ontologies, but so far they have been restricted either by structure, by limited inheritance rules, or by interoperability with ontologies designed by another designer. For an ontology to be truly flexible and useful, it must contain all of deduction, relations among first-class objects, comparison or merging and subsumption (or some type of flexible inheritance).

Gruber [5] proposes a list of design criteria for ontologies as a method for evaluating an ontology. While these criteria help to clarify the issues behind ontology design, and help to guide the ontology designer, they are not formal, testable definitions of an ontology. It was not Gruber's intention to offer formal definitions for ontologies. Gruber sees an ontology as a statement of a logical theory, but eschews the idea that ontologies might be limited to traditional logical definitions that are simply contentless logical terms without any world knowledge. Then agents are said to have an *ontological commitment* to an ontology when the agent's observable actions are consistent with the definitions of the ontology [5].

The problem, however, is that there is no way in Gruber's system to guarantee or enforce the commitment. For example, while Gruber's guidelines say that an ontology should be coherent and consistent, how do you guarantee consistency and coherence without a formal guarantee of soundness?

The extensive work by McGuiness and others [6-8] provides many tools for using, manipulating and even comparing ontologies, but still fall short in being completely expressive. The Chimaera ontology environment, for example, provides a tool for comparing knowledge bases created by different designers, and is effective at finding objects common to two ontologies. However, Chimaera is completely user-driven, in that no automated merging of the ontologies is produced by the tool. While this provides a useful tool for many types of applications, there is no automation in the ontology comparison, except to attempt to identify common words from two ontologies. Further, much of this work is restricted to taxonomies, and not generalized lattice structures, which restricts the domains that can be explored.

OWL [9] (and the language it was derived from, DAML+OIL) is another ontology tool which is based on Description Logics [7]. Fundamentally, DAML+OIL and OWL represent a set of syntax rules built on a decidable fragment of First Order Logic (FOL). The semantics of the language are derived from its FOL foundations, so there are useful definitions of properties such as deduction and soundness. (However, there was a proposed model-theoretic semantics for DAML+OIL[1].) However, when using the most common ontology tools, many authors have trouble when describing the relation between a class and an object [10]. For example, DAML+OIL makes a clear separation between object classes and data types [11]. Another major criticism of DAML+OIL was that it had a "weak semantics," an issue that still has not been fully addressed in OWL [12]. Neither language supports a flexible inheritance.

We must conclude that while there has been a large body of research work generated around the design, use, and reuse of ontologies, many authors seem to be converging on the idea that a formal definition of the semantics of ontologies is now becoming necessary. This semantics must include deduction, relations among first-class objects, comparison or merging of objects and classes, and subsumption (or some type of flexible inheritance) as discussed previously.

3 Conceptual Graphs as Ontology Representation

The claim in this work is that conceptual graphs have the appropriate expressiveness for the task of representing, comparing, merging and interoperating with ontologies. While it is true that languages and formalisms like Description Logics, OWL, DAML+OIL and others can do similar work, Conceptual Graph Theory (CGT) has the tools, structure and formal definitions already in place to handle the functional requirements mentioned above. CGT uses the same reasoning, subsumption, inheritance and comparison mechanisms for types as for individuals. This means that there

[1] See http://www.daml.org/2001/03/model-theoretic-semantics.html

is no difficulty in using types as objects, or in manipulating types in a reasoning process.

Our further claim is that Conceptual Graph Theory is the best methodology to use for the purpose of defining a semantics for ontologies. A CGT-defined ontology will give the developer some of these functionalities automatically, and has the added benefit of well-defined type hierarchies, semantics and subsumption. The claim is that in order to tailor information/knowledge to a user, having a CGT ontology is a very good, well-defined way of creating a filter through which a user can specify the knowledge that they're looking for, and what should be left out.

Previous work in using CGT to represent ontologies has been undertaken. In [13] the authors describe an algorithm to automatically translate RDF schemas into conceptual graphs. While this work demonstrates the compatibility of some semantic web schemes and CGT, it doesn't utilize the powerful semantics of CGT. Since the RDF is just the syntactic structure of a knowledge base, translating these into conceptual graphs is straightforward, as there is no need for a semantic representation.

The rest of this article will concern itself with methods for representing ontologies using tools and techniques which are already well-defined, and can be applied to ontologies. Fundamentally, this work is about laying the theoretical foundations for knowledge base interchange, reuse and filtering. We now proceed to discuss CGT, and the elements of CGT that can be used to implement ontology operations, tailoring and filtering.

4 A Formal Definition of Ontology

The formal definitions of conceptual graphs and CG canon have been presented many times, and so there is no need to introduce the formal definitions again here. Suffice it here for us to recall that formally a domain for CGs is described by a canon which is composed of the set T of types, the set I of individuals (variables or constants), a subtype relation which gives a partial ordering of the types of the individuals, the conformity relation :: which relates type labels to individuals $\in$ I and the canonical basis function B (also called σ by some authors) which associates each relation type with the concept types that may be used with that relation, which helps to guarantee well-formed graphs.

The set T of types is arranged into a type hierarchy, ordered according to the specificity of each type. Separate type hierarchies are established for the concepts and the relations within a canon. The hierarchy is expressed by a subsumption or generalization order on the types. A type t is said to be more specific than a type s if t specializes some of the concepts from s. Projection is the function which determines the specialization/generalization relation between two concepts.

The definitions for type hierarchies and for the operations on those hierarchies inform our definition of an ontology. We want an ontology to provide a framework for the semantics of a domain. A canon, as defined in CGT, provides the background for the representation, since we can use the definitions of relations, subsumption and conformity to support our definition of an ontology. We can now formally define an ontology as the particular set of hierarchies that are created for a given domain, along with all of the operations on a canon.

Definition 1. Ontology. An ontology in a given domain M with respect to a canon is a tuple (T_{CM}, T_{RM}, I_M) where

T_{CM} is the set of concept types for the domain M and T_{RM} is the set of relation types for the domain M.

I_M is the set of individuals for the domain M.

An ontology is then a collection of types and individuals, which forms a framework for the knowledge in a domain. The collection is arranged into a hierarchy based on the subtype relation $\leq$. The canon provides the basis for subsumption in the ontology and guarantees consistency among the relations and in the typing of individuals.

Note that this hierarchy is not necessarily a taxonomy, in that a type may have multiple supertypes. Further note that there is no point on the hierarchy where we must make a distinction between a type and an instance. Every concept on the hierarchy is treated as a type. A type may have subtypes and supertypes, but there is no need to distinguish these from instances of the types.

This is distinct from the object-oriented objective of objects inheriting all the properties of a class of objects. The essential difference is in, for example, treating a kitchen as you would any generic room. The *type* room can be placed, occupy space, and have specific values for color and number of doors. A *class* of rooms will have attributes, but cannot be said to occupy a space or have specific dimensions, or have a specific count or placement of doors. The generic room can have constraints placed on its attributes, and finally can be specialized into a kitchen. Fundamentally, a generic room can take the place of a specialized room, unlike a class of objects.

The ontology (as a concept type hierarchy) acts as the framework, with conceptual graphs that conform to the hierarchy used to instantiate concepts in the domain. The ontology is populated by creating conceptual graphs which represent actions, ideas, situations or states in the domain. Recall, though, that a conceptual graph need not be a complete description, and will always be treated in the same manner as any other type.

The closest approaches to demonstrating an equivalence between FOL and Conceptual Graphs are due to Chein and Mugnier [14] and to Amati and Ounis [15]. They use a restrictive form of CGs, in which each concept type is allowed only one individual to represent it. Once the existential operator has been applied to a generic referent, all concepts of that type must use that one individual. Clearly, this makes it much easier to interpret Conceptual Graphs into FOL. Given that restriction, Amati and Ounis show that graph derivation through projection is sound and complete. They discuss a method for graph deduction on these restricted graphs.

The real significance of the work by Chein and Mugnier, Amati and Ounis, and indeed of our own work, is the proof that deduction systems over Conceptual Graphs are not only possible, but also effective ways of handling knowledge merging, comparison and reasoning.

5 Projection of Ontology Types

The definitions of consistency and type subsumption in this paper are based on formal concepts of projection and lower bounds from Conceptual Graph Theory [16]. Projection is the operation used to determine subsumption relations, and to find similarities between parts of the knowledge base. A more general type *G* is said to subsume a more specific type *H* if *G* has a projection into *H*. For example, the type *mammal* would have a projection into the type *cat*.

The following definitions of projection are modified from the standard definition used in recent Conceptual Graph literature [4, 17-19]. Rather than defining projection from one graph into another, these definitions represent projection of types, and therefore defines the subsumption operator on type hierarchies.

Definition 2. Concept projection. Given two concept types, $\boldsymbol{s} \in \mathrm{C}$ and $\boldsymbol{t} \in \mathrm{C}'$, *s* is said to have a projection into *t* if and only if there is a morphism $\boldsymbol{h_C}$: $\boldsymbol{C \rightarrow C'}$, such that:

$\forall s \in C$ and $\forall t \in C'$, $h_C(s) = t$ only if *type(s)*   *type'(t)*, and ***referent*(*s*)** $= *$ or *referent(s) = referent'(t)*

C is the set of concepts, *type* : $\boldsymbol{C \rightarrow T}$ indicates the type of a concept, and *referent* : $\boldsymbol{C}$ $\rightarrow \boldsymbol{I}$ indicates the referent marker of a concept.

Definition 3. Relation projection. Given two relation types, *r* and *r'*, *r* is said to have a projection into *r'* if and only if there is a morphism $\boldsymbol{h_R}$: $\boldsymbol{R \rightarrow R'}$, such that:

$\forall r \in R$ and $\forall r' \in R'$, $h_R(r) = r'$ only if *type(r)*   *type'(r')*
R is the set of relations, and *type* : $\boldsymbol{R \rightarrow T}$ indicates the type of a relation.

The definition of type subsumption is based on notions of graph projection. Projection and subsumption are defined for individual graphs to help determine their ordering in accordance with the type hierarchy, and to allow unification, deduction and combination of graphs. While we concern ourselves here with issues of type projection, the topic of graph projection and subsumption is covered in detail in [18] and [20].

This definition of projection then gives us a formal definition for subtype and supertype and for subsumption on the partial order of the types in the hierarchy. The operations of join, meet and unify are now simply applications of the projection operator. Finding types which are compatible (i.e. that can be unified) is now a matter of finding a common subtype (or *join*) between the two types. If the only common subtype is $\perp$ then there can be no comparison.

Consistency and validity of conceptual graphs are guaranteed by adhering to strict rules regarding the formation of new graphs. A domain is defined by a set of basic graphs, the canonical basis and by its type hierarchies. All other graphs must be derived from the canonical basis by the use of the canonical formation rules. Like the rest of the CG field, these rules are still evolving, but they all involve the same basic ideas, as expressed here.

The significance of the canonical formation rules is that they specify the mechanisms for deriving new graphs and types. These are the techniques used to implement the formalisms defined earlier. We now have a complete formalism for the specification and manipulation of knowledge bases expressed using ontologies. If an ontology author is careful to set up the ontology according to the definitions of subsumption and projection, and then follows the canonical formation rules when deriving new types, expressions and graphs, the result will always be graphs and extensions to the ontology which are canonical according to the domain. If CG unification and reasoning are applied (as defined in [20]) then the derived graphs and types will also be valid in the domain.

6 Ontology Operators as Mergers of Knowledge

As an operator for ontology filtering and comparison, the use of the projection operator becomes obvious. A user would implement an ontology as a type hierarchy locally. Whenever the user sends a query to a database or knowledge base, or performs a web search, the user's ontology is compared with the ontology which has (hopefully) been implemented by the designer of that knowledge base. The query is examined in light of the local ontology and the remote ontology, and information from the remote knowledge base is filtered and constrained based on the comparison.

This technique can be used for two practical purposes: comparing and merging ontologies, or to inform and filter queries. We examine examples of each of these below.

6.1 Ontology Comparison and Merging

When comparing or merging the knowledge from two ontologies, one need only determine whether pairs of concepts under consideration (one from each ontology) are in a subtype-supertype relation. In order to facilitate this process, and to prevent the complexity of a general graph match problem, we restrict our ontologies to a finite number of nodes and types, and we insist that the user must specify a starting node for each ontology being compared. This means that there needs to be a way for specifying which two types to compare. This may simply be T, but can be any node that the user wants to specify. Comparison of the two ontologies then proceeds from the T node in a depth-first manner through the graph of the hierarchy.

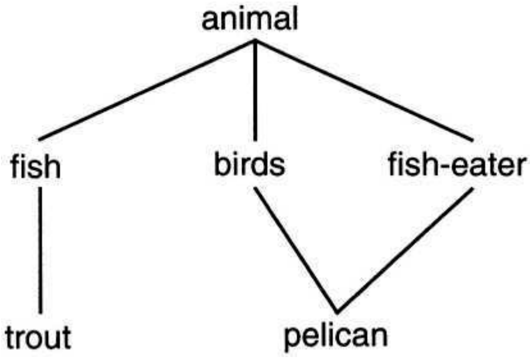


Fig. 1. A type hierarchy.

Reynolds, in his work in defining unification [21], used the natural lattice structure of first-order terms, which was a partial ordering based on subsumption of terms [22]. Many terms (or types in our case) are not in any subsumption relation, for example *cat* and *dog,* or *wood* and *mammal.* Inheritance hierarchies can be seen as lattices that admit unification and generalization [23]. So, in our case, combining two ontologies is the process of finding the common points in the two ontologies (represented as lattices) and merging the rest of the structures together, in a similar manner to the unification of two CGs.

We take our example of ontology merging from a constraint approach in H. Aït-Kaci's Login [24]. The novel contribution of **ψ-terms** is in the use of type inheritance information. Aït-Kaci's view of unification was as a filter for matching partial structures, using functions and variables as the "filters". Aït-Kaci allowed type information to be attached to functions and variables. Then, his unification technique uses information from a taxonomic hierarchy to achieve a more gradual filtering.

An example of Aït-Kaci's ideas from Knight [23] illustrates our merging technique. Assume that we have the following inheritance information, as illustrated in Figure 1: Birds and fish are animals; a fish-eater is an animal; a trout is a fish; and a pelican is both a bird and a fish-eater. Then unifying the following **ψ-terms**:

fish-eater (likes → trout)
bird (color → brown; likes → fish)

will yield the new **ψ-term**:

pelican (color → brown; likes → trout)

Unification does not fail on comparing *fish-eater* to *bird,* or *trout* to *fish.* Instead, the conflict is resolved by finding the greatest lower bound on each of the two pairs of items in the taxonomic hierarchy, in this case *pelican* and *trout,* respectively. In this manner, Aït-Kaci's system naturally extends the knowledge-merging (or *knowledge conjunction*) nature of unification.

The formal definition of unification for Conceptual Graphs is set out in earlier work [18, 25]. However, in our case, rather than examining the unification of two conceptual graphs, we want to explore the knowledge conjunction of two ontologies, or type hierarchies.

The merging of two ontologies is somewhat more complicated, and also more interesting and useful than merely an extension of the join and unification operations. The unification of two graphs contains neither more nor less information than the two graphs being unified. The merging of two ontologies is a matter of finding a common starting point on the two hierarchies (usually with the assistance of the user) and then continuing outward from that point in a depth-first manner to find other matching points.

The main thrust of previous research has been the unification of Conceptual Graphs in terms of conjoining the knowledge contained in two different graphs [26, 27]. In our case, these pieces of partial information are represented by Conceptual Graphs. However, our current work involves combining the knowledge of two entire domains. We want to be able to combine the expert knowledge of two systems, or even combine knowledge from different sources, not merely gather additional information.

We have defined unification as the combining of pieces of knowledge in a domain, represented as Conceptual Graphs. We define unification as an operation that simultaneously determines the consistency of two pieces of partial or incomplete knowledge, and if they are consistent, combines them into a single result.

When an ontology is represented by the use of Conceptual Graphs constructed in this way, subsumption can be used to combine, refine and reuse the knowledge contained in the graphs. This further allows us to perform reasoning over the knowledge in the graphs as concepts. Reasoning is not limited to objects, classes or libraries, but can also be applied to generic concepts in the knowledge.

So, this method involves creating a completely new ontology by merging the two ontologies. Taking for our example Figure 1, which shows a small ontology of animals, we find another ontology with similar knowledge which could inform and expand our first ontology. This is shown in Figure 2.

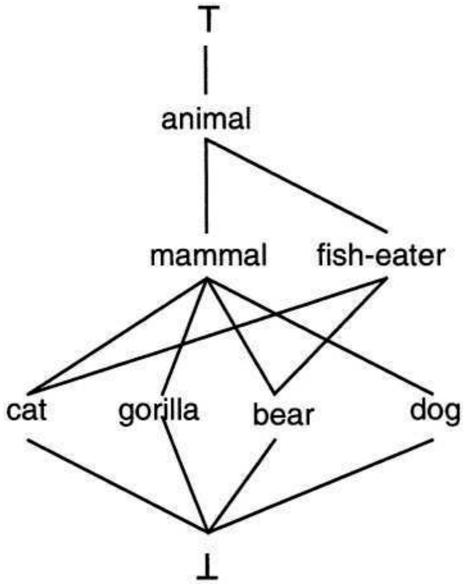


Fig. 2. Another ontology from the same domain.

We can now employ standard graph spanning and unification algorithms which recursively move down through the subsumption relations from the start node. Informally, our algorithm starts by comparing the start nodes, q and q′ for compatibility, that is, that they are the same type or that the type of one subsumes the type of the other. Given compatible start nodes, the algorithm selects a subsumption relation s from all of the subsumption relations that lead from the head node q, and seeks its projection s′ in the second lattice from the subsumption relations that lead from q′. If none is found, then the subsumed type s becomes part of the unified graph trivially. If the projection is present in the second graph, then the unification algorithm is called recursively on the types subsumed by type s. If all these types s prove to be compatible, then the two sublattices are joined, and attached to the unified top, q″. The algorithm proceeds depth-first through the lattice. When all types subsumed by the start concept in both graphs have been successfully processed in this way, the algorithm terminates successfully. Figure 3 shows the result of applying the Merge algorithm to the lattices in Figs. 1 and 2.

6.2 Query Filtering with Ontologies

The second major use of ontologies is to filter or select information from web queries into a knowledge base. The concept can be illustrated by assuming two ontologies created by separate users. One ontology relates knowledge about the user's interests in research, and shows that this user has an interest in agents and in robots. This user considers a mobile agent to be both an agent and a robot, and separates web agents and demons from the ideas of mobile agents.

This user wants to find others who are interested in similar areas, with the idea of locating potential collaborators, learning from their work, and downloading information from their web sites. The user sends a query by way of a web agent to find a match for other ontologies. In this case, the user has specified that a match consists

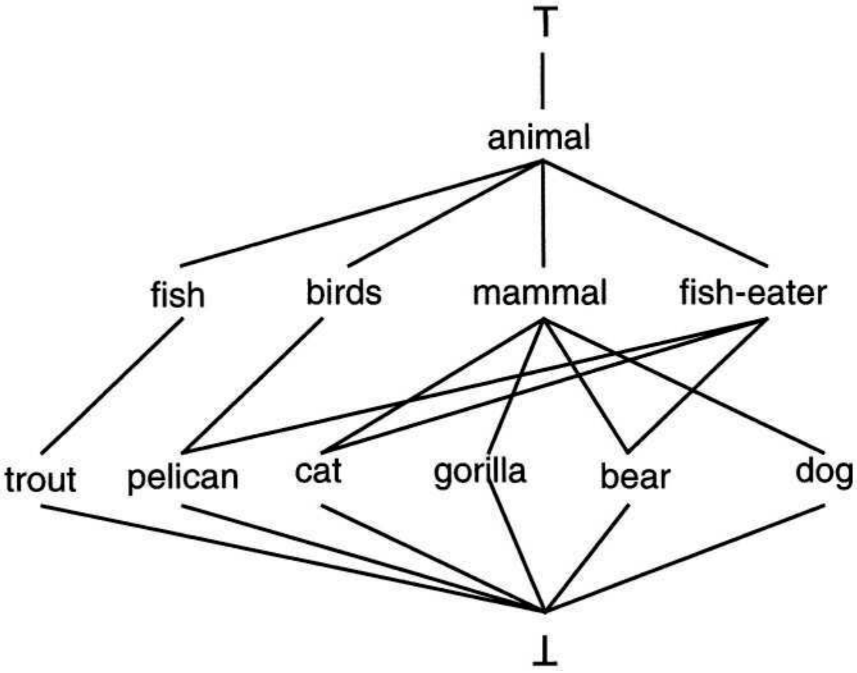


Fig. 3. The merged ontology.

of the use of the type *robot* in a subsumption relation with the type *mobile agent.* The query is sent as a fragment of the graph which represents the user's ontology, as shown in Fig. 4. (We elide over the processes of the web agent and the sending of the query, as there already exist many techniques for this.)

The second ontology shows a user with a research interests in robots, but where the ontology designer has put an emphasis on dividing the work into hardware and software components, shown in Fig. 5. A match is found between the two ontologies, since in both cases we find the type *robot* subsuming the type *mobile agent.* The user would then be able to examine any types and concepts subsumed by mobile agent for items of interest.

7 Conclusions and Future Work

The method described above works well when the user can guarantee that a given type name has the same meaning in all ontologies. However, problems arise when a given label has different meanings in two ontologies, or when the semantics of the types vary. This is the classic problem presented by many ontology authors [3, 28]. Overcoming the problems of synonymy and heterogeneity requires the examination of the semantic links of each type. In our work, this means comparing the semantic links of each type in the "neighborhood" of each type, and requires a basic understanding of how to compare the neighboring nodes for semantic similarity.

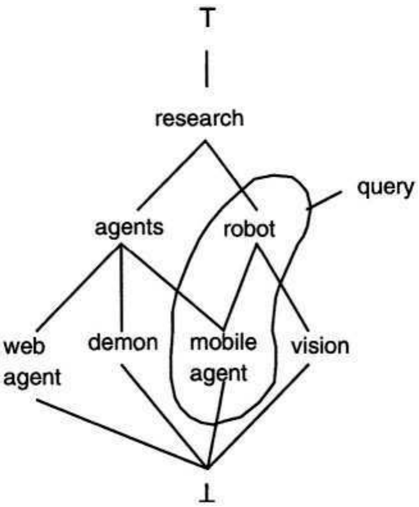


Fig. 4. A user's ontology and query.

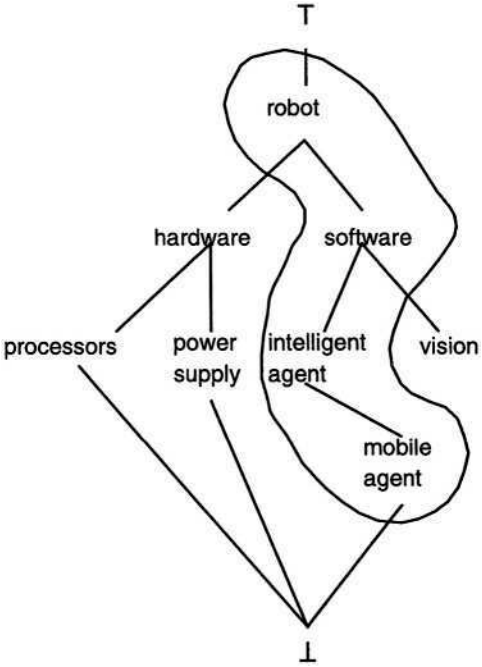


Fig. 5. Another ontology.

Fundamentally, ontology comparison and merging for the purposes of tailoring and filtering the information provided by knowledge servers is a hard problem. Some authors have explored solutions that include DAML+OIL and other languages specific to ontology. Other authors have explored semantic closeness measures for ontologies which imitate the lattice structures of the CGT formalism [28], which allow a comparison of ontologies. As discussed by Arara and Benslimane, ontologies are more than just another database formalism, since they must also include attributes, relations, and functionalities (or some sort of interaction with the real world) [29].

We have demonstrated a method for automated comparison of ontologies represented as concept type hierarchies. Type hierarchies and the canonical formation rules efficiently specialize ontologies into concrete instances by instantiating canonical CGs from the hierarchies. A simple merge operation, using join and type subsumption, is used to perform knowledge conjunction of the concepts represented in the ontologies. The significance of our work is that the previously static knowledge representation of ontology is now a dynamic, functional knowledge provider/server. This extension of ontologies using Conceptual Graph Theory helps to strengthen the use of ontologies as a knowledge representation and reasoning system by providing projection and subsumption, and by eliminating the boundary between class and object. The work presented in this paper is a first step toward formally defining a method for comparing the semantics of two objects, based on their position in a type hierarchy.

This work is also significant for the fact that we have described a system, using well-understood techniques, for comparing, combining and merging ontologies. The ontologies being compared can be from separate domains, similar domains or even ontologies which were created by different authors.

References

1. Mineau, G. “A First Step Toward the Knowledge Web: Interoperability Issues Among Conceptual Graph Based Software Agents, Part I”, in *Proc. International Conference on Conceptual Structures.* 2002. Borovets, Bulgaria: Springer-Verlag. Published as LNAI volume 2393.
2. Sowa, J.F., *Negotiation Instead of Legislation.* 2002. Available at: www.jfsowa.com/talks/negotiat.htm
3. van Zyl, J.D. and D.R. Corbett. “A Framework for Comparing Methods for Using or Reusing Multiple Ontologies in an Application”. in *Proc. Eighth International Conference on Conceptual Structures.* 2000. Darmstadt, Germany: Shaker Verlag.
4. Mugnier, M.-L. and M. Chein, “Représenter des Connaissances et Raisonner avec des Graphes”. *Revue d’Intelligence Artificielle,* 1996. **10**(6): p. 7-56.
5. Gruber, T.R., “Toward Principles for the Design of Ontologies Used for Knowledge Sharing,” in *Formal Ontology in Conceptual Analysis and Knowledge Representation,* N. Guarino and R. Poli, Editors. 2003, Kluwer Academic Publishers. Substantial revision of paper presented at the International Workshop on Formal Ontology, March, 1993, Padova, Italy.
6. McGuinness, D.L., “Ontologies Come of Age,” in *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential,* D. Fensel, et al., Editors. 2003, MIT Press.
7. McGuinness, D.L., R. Fikes, J. Hendler, and L.A. Stein, “DAML+OIL: An Ontology Language for the Semantic Web”. *IEEE Intelligent Systems,* 2002. **17**(5): p. 72-80.
8. McGuinness, D.L., R. Fikes, J. Rice, and S. Wilder. “The Chimaera Ontology Environment”. in *Proc. Seventeenth National Conference on Artificial Intelligence.* 2000. Austin, Texas, USA.
9. Dean, M., D. Connolly, F.v. Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein, *OWL Web Ontology Language 1.0 Reference.* 2002. Available at: http://www.w3.org/TR/owl-ref/
10. Bechhofer, S., L. Carr, C. Goble, S. Kampa, and T. Miles-Board. “The Semantics of Semantic Annotation”. in *Proc. International Conference on Ontologies, Databases and Semantics.* 2002. Irvine, California, USA: Springer. LNCS #2519.
11. Heymans, S. and D. Vermeir. “A Defeasible Ontology Language”. in *Proc. International Conference on Ontologies, Databases and Applications of Semantics.* 2002. Irvine, California, USA: Springer Verlag. Published as LNAI volume 2519.
12. Smith, M.K., *Web Ontology Issues.* W3C, 2002. Issue# 5.10
13. Corby, O., R. Dieng, and C. Hébert. “A Conceptual Graph Model for W3C Resource Description Framework”. in *Proc. Eigth International Conference on Conceptual Structures.* 2000. Darmstadt, Germany: Springer-Verlag. LNAI #1867.
14. Chein, M. and M.-L. Mugnier, “Conceptual Graphs: Fundamental Notions”. *Revue d’Intelligence Artificielle,* 1992. **6**(4): p. 365-406.
15. Amati, G. and I. Ounis, “Conceptual Graphs and First Order Logic”. *The Computer Journal,* 2000. **43**(1): p. 1-12.
16. Sowa, J.F., “Conceptual Graphs Summary,” in *Conceptual Structures: Current Research and Practice.* 1992, Ellis Horwood: Chichester, UK.
17. Willems, M. “Projection and Unification for Conceptual Graphs”. in *Proc. Third International Conference on Conceptual Structures.* 1995. Santa Cruz, California, USA: Springer-Verlag. Published as Lecture Notes in Artificial Intelligence #954.

18. Corbett, D.R., "Conceptual Graphs with Constrained Reasoning". *Revue d'Intelligence Artificielle,* 2001. **15**(1): p. 87-116.
19. Leclère, M. "Reasoning with Type Definitions". in *Proc. Fifth International Conference on Conceptual Structures.* 1997. Seattle, Washington, USA: Springer-Verlag. Published as Lecture Notes in Artificial Intelligence #1257.
20. Corbett, D.R., *Reasoning and Unification over Conceptual Graphs.* 2003, New York: Kluwer Academic Publishers.
21. Reynolds, J.C., "Transformational Systems and the Algebraic Structure of Atomic Formulas". *Machine Intelligence,* 1970. **5**.
22. Davey, B.A. and H.A. Priestley, *Introduction to Lattices and Order.* 1990, Cambridge: Cambridge University Press.
23. Knight, K., "Unification: A Multidisciplinary Survey". *ACM Computing Surveys,* 1989. **21**(1): p. 93-124.
24. Aït-Kaci, H. and R. Nasr, "LOGIN: A Logic Programming Language with Built-in Inheritance." *Journal of Logic Programming,* 1986. **3**.
25. Corbett, D.R. "A Framework for Conceptual Graph Unification". in *Proc. Eighth International Conference on Conceptual Structures.* 2000. Darmstadt, Germany: Shaker Verlag.
26. Corbett, D.R. and R.F. Woodbury. "Unification over Constraints in Conceptual Graphs". in *Proc. Seventh International Conference on Conceptual Structures.* 1999. Blacksburg, Virginia, USA: Springer-Verlag.
27. Corbett, D.R. "Reasoning with Conceptual Graphs". in *Proc. Fourteenth Australian Joint Conference on Artificial Intelligence.* 2001. Adelaide, South Australia: Springer. Published as LNAI volume 2256.
28. Buccafurri, F., G. Lax, D. Rosaci, and D. Ursino. "A User Behavior-Based Agent for Improving Web Usage". in *Proc. International Conference on Ontologies, Databases and Applications of Semantics.* 2002. Irvine, California, USA: Springer-Verlag. LNAI #2519.
29. Arara, A.A. and D. Benslimane. "Ontology Concept Extraction from Terminologies". in *Proc. Eleventh International Conference on Intelligent Systems.* 2002. Boston, Mass, USA: ISCA.

A Priorean Approach to Time Ontologies

Peter Øhrstrøm and Henrik Schärfe

Department of Communication, Aalborg University, Denmark
{poe, scharfe}@hum.aau.dk

Abstract. Any non-trivial top-level ontology should take temporal notions into account. The details of how this should be done, however, are frequently debated. In this paper it is argued that “the four grades of tense-logical involvement” suggested by A.N. Prior form a useful framework for discussing how various temporal notions are related in a top-level ontology. Furthermore, a number of modern ontologies are analysed with respect to their incorporation of temporal notions. It is argued that all of them correspond to Prior’s first and second grade, and that none of them reflect the views which Prior’s third and fourth grade represent. Finally, the paper deals with Prior’s ideas on a tensed ontology and it is argued that a logic based on the third grade and will be useful in the further development of tensed ontology.

1 Introduction

In post-Medieval logic ontology, conceived as the study of being or existence, was regarded as a very important field of study. Indeed, philosophers and logicians in the 17th and 18th century were the first to establish the very word ‘ontology’ as a technical term naming the study of being or existence as such. One of the first philosophers to do so was Christian Wolff (1679-1754) who wanted to construct a logical system with which he could account for existence and non-existence. According to Wolff a ‘being’ or an ‘entity’ is whatever can be thought without involving any logical contradiction. This means that Wolffian ontology does not answer the question regarding what there is in the actual world. This question has to be answered within the so-called special metaphysics. In the 20th century, most philosophers interested in ontology have accepted the logical approach to ontology, but they have not been satisfied with the Wolffian notion of existence. In contrary, they mainly wanted to discuss the actual world, and not only the possible, but non-real worlds. Stanislaw Leśniewski (1886-1939) seems to have been the first logician to suggest a formal (deductive) calculus of ontology. The core of this contribution is mereology which deals with the part–whole relation between objects. In the 1950s another Polish logician, C. Lejewski, extended the Leśniewskian calculus introducing a so-called ‘chronology’, according to which objects are conceived as corresponding to durations.

In the works of Edmund Husserl (1859-1938) and Martin Heidegger (1889-1976) there is a strong emphasis on ideas of time as a very important background for the study of ontology. This means that there is an essential relation between ontological inquiry and the study of the concept(s) of time (see [8]). In his philosophical logic

K.E. Wolff et al. (Eds.): ICCS 2004, LNAI 3127, pp. 388-401, 2004.

A.N. Prior (1914-69) put a similar emphasis claiming that temporal aspects are crucial for any satisfactory approach to reality.

Willard Van Orman Quine (1908-2000) was one of the most influential 20th century philosophers who contributed to the further development of ontology. In his "On What There Is" [23]. Quine introduced a view of ontology as the study of the ontological commitments of natural science (see [26]). According to Quine's view the ontologist should derive a network of claims about what actually exists using the natural sciences as his main source. In addition the ontologist should attempt to establish what types of entities are most basic. For Quine first-order predicate logic is crucial in the determination of what actually exists. According to his so-called 'criterion of ontological commitment' the ontologist can conclude that the ontological commitment of a scientific theory implies the existence of certain entities (such as electrons) if the theory presupposes that these entities are values of bound variables when formulated in first-order predicate logic.

A.N. Prior found Lejewski's as well as Quine's ideas on ontology very challenging, and he offered an idea of ontology which may be seen as an alternative to both approaches. Prior accepted the importance of Lejewski's attempt to involve time in the description of objects, but he found that the incorporation of time had to be carried out in a different manner. Prior's accepted Quine's idea that the ontologist should try to establish what types of entities are most basic, but he rejected the Quinean idea of letting ontology depend on what kinds of variables we are prepared to bind by quantifiers. In his writings Prior produced a highly original way of dealing with the temporal aspects of reality and the notions of quantification. In fact Prior suggested four conceivable theories regarding the temporal aspects of reality (the so-called four grades of tense-logical involvement). In section 2 we are going to describe these theories in some details and we shall see how they allow for different scopes of quantification. Each of the four theories suggests an ontology explaining how the various temporal notions and concepts (such as 'past', 'present', 'future', 'before', 'after', 'instant', 'duration') are mutually related. Although Prior stated that all four theories are possible, he himself clearly favoured the last of them (i.e. the 4th grade of tense-logical involvement).

For at least two decades formal ontologies have been studied intensively within various parts of computer science (see Sowa [27], [28]). Knowledge representation is obviously one of the computer science disciplines in which formal ontology has turned out to be particularly important. According to T.R. Gruber [10] the term 'ontology' is used in this field as standing for *»a specification of a conceptualization«,* or to put it differently »a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents«. This may be said to bring us back to the Wolffian notion of existence according to which an entity exists if it can be thought of without any contradiction.

During recent years one of the most influential ideas within formal ontology has been the idea of a top-level ontology, which should include a small number of highly general categories. The benefits of such a top-level ontology is obvious when it comes to the development and application of information systems. Although very few believe that it will be possible ever to reach a universal agreement on the formulation of a non-trivial top-ontology, everybody working with information systems will accept the view that the problem of classification is crucial for the formulation of such systems. For this reason, the debates on formal ontology and classification should go on, and everybody working with information systems and knowledge engineering should be aware of the ontological challenges to which the problems of concept handling give rise.

Prior did not pursue Lejewski's idea of a durational logic. However, other workers in temporal logic have done so. Their works show that the ontological relations between durations and instants are by no means given. Section 3 deals with the construction of formal ontologies in the light of the works within durational logic.

In section 4 we shall discuss the analysis of temporal notions according to some of the most well-known top-level ontologies, and we shall argue that most authors of modern ontologies, apparently without much reflection, assume something like Prior's first or second grade of tense-logical involvement. In section 5 we intend to discuss these Priorean ideas of a tensed ontology. We shall see that there is still much to be done in order to carry out the full Priorean program.

2 Prior's Four Grades of Tense-Logical Involvement

In his famous paper 'The Unreality of Time' [15] John Ellis McTaggart (1866-1925) suggested a distinction between A- and B-series, which in fact corresponds to a distinction between the following two sets of temporal notions:

A-notions: past, present, future
B-notions: before, after, 'simultaneous with'

Since McTaggart, philosophers and logicians have discussed intensively which of the two sets is the ontologically more fundamental one for the philosophical description of time. During the 20th century several authors have tried to answer this question of temporal ontology in a formal manner. In fact, this discussion has turned out to be essential for the formulation of temporal logic. A.N. Prior (1914-69) who was the founder of modern temporal logic paid very much attention to this problem. In his brilliant and very important paper from 1968, 'Tense Logic and the Logic of Earlier and Later' [21: 117ff.], Prior introduced four so-called grades of tense-logical involvement. These four grades can be understood as four different ways in which the question of temporal ontology can be answered.

The first grade defines tenses entirely in terms of objective instants and an earlier-later relation. For instance, a sentence such as *Fp,* 'it will be the case that *p*', is defined as a short-hand for 'there exists some instant *t* which is later than now, and *p* is true at *t*', and similarly for the past tense, *Pp,* i.e.

(DF) $T(t,Fp) \equiv_{def} \exists t_1: t< t_1 \wedge T(t_1,p)$

(DP) $T(t,Pp) \equiv_{def} \exists t_1: t_1< t \wedge T(t_1,p)$

Given that $G \equiv_{def} {\sim}F{\sim}$ ('it is always going to be the case that ...') and that $H \equiv_{def} {\sim}P{\sim}$ ('it has always been the case that...') it easily follows that

(DG) $T(t,Gp) \equiv \forall t_1: t< t_1 \supset T(t_1,p)$

(DH) $T(t,Hp) \equiv \forall t_1: t_1< t \supset T(t_1,p)$

According to this view, tenses can be considered as mere meta-linguistic abbreviations, so this is the lowest grade of tense logical involvement. Prior succinctly described the first grade as follows:

> ...there is a nice economy about it; it reduces the minimal tense logic to a by-product of the introduction of four definitions into an ordinary first-order theory, and richer [tense logical] systems to by-products of conditions imposed on a relation in that theory. [21: 119]

In the first grade, the tense operators *(P, F, G,* and *H)* are simply tools with which we can establish a handy way of summarizing the properties of the before-after relations, which constitute the B-theory of McTaggart. Hence, in the first grade temporal instants are viewed as something primitive and objective. Together with the before-after-relation they are seen to be determining for a proper understanding of time and reality. According to this view tenses are deemed to have no independent ontological status. The same can be said about the modal operators, $\Box$ and $\Diamond$, which correspond to necessity and possibility, respectively. These operators can be defined in the following way:

$$\Box p \equiv_{def} \forall t: T(t,p)$$
$$\Diamond p \equiv_{def} \exists t: T(t,p)$$

The time ontology of the first grade can be presented in the following way:

1st grade	
Primitive notions	Instants. The before-after relation (<). The truth-function ('true at an instant'), *T(t,p)*. First order logic.
Derived notions	Tenses: past (*P*), always past (*H*), future(*F*), always future(*G*). Modalities: necessity ($\Box$), possibility ($\Diamond$).

This time ontology seems to be the dominating and most common way of understanding the temporal aspect of reality. The weaknesses of this approach are, however, rather obvious. The most important problem is that the first grade does not include any idea of 'now'. In this system the present time can only be represented as an arbitrary instant. This is of course quite acceptable, if we take the view which Albert Einstein expresses in a letter to Michele Besso in the following way:

> There is no irreversibility in the basic laws of physics. You have to accept the idea that subjective time with its emphasis on the now has no objective meaning. [Quoted from [18: 203]]

If the 'Now' and consequently also the other A-concepts are purely subjective, all we have to bother with when dealing with the objective world are the B-series.

Viewed in this way reality is just a four-dimensional co-ordinate system. Times are nothing but clock-readings and dates. Prior described this position in the following way:

> Whether the events are the case or merely have been or will be, is of no concern to the scientist, so he uses a language in which the difference between being, having been, and being about to be becomes inexpressible. [20: 323]

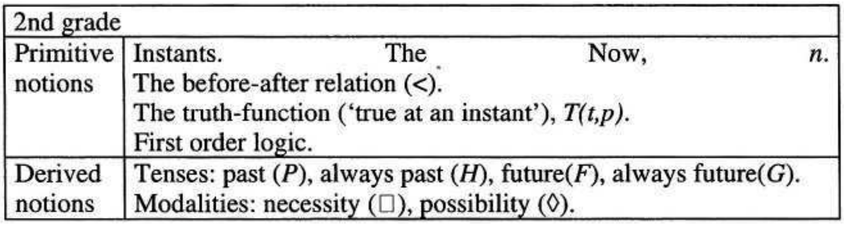
However, Prior has argued that the following shows that even Einstein was a bit uncertain regarding the status of the 'Now':

> Einstein himself once said to Carnap that the problem of the Now worried him seriously. He explained that the experience of the Now means something special for men, something different from the past and the future, but that this important difference does not and cannot occur within physics. [21: 136-137]

Prior, himself, insisted that 'the Now' is real, and consequently that the distinction between the tenses is real (see [9: 47]). In his view it would be mistaken to assume that the 'Now' and the tenses can be derived from the logic of earlier and later. For this reason he had to reject the first grade and turn to an ontology in which the tense-logical aspect of reality is assumed to be fundamental.

In the second grade of tense logical involvement, the 'Now' is in fact accepted as a primitive notion. Formally, in this theory 'the Now' can be treated as a prepositional constant, *n*. This means that according to this view A- and B-concepts are treated on a par. Specifically, a bare proposition *p* is treated as a syntactically full-fledged proposition, on a par with what Rescher and Urquhart [24] called 'chronologically definite' propositions such as *T(t,p)* ('it is true at time *t* that *p*'). The point of the second grade is that a bare proposition with no explicit temporal reference is not to be viewed as an incomplete proposition. One consequence of this is that an expression such as *T(t,T(t',p))* is also well-formed, and of the same type as *T(t,p)* and *p*. In fact, p can be understood as being equivalent with *T(n,p)*. i.e. *p* is true if and only if *p* is true now. Prior showed how such a system leads to a number of theses, which relate tense logic to the earlier-later calculus and vice versa [19: 119].

The philosophical implication of this second grade of tense logical involvement is that one must regard the basic A- and B-theory concepts as being on the same conceptual level. Neither set of concepts is conditioned by the other.



2nd grade	
Primitive notions	Instants. The Now, *n*. The before-after relation (<). The truth-function ('true at an instant'), *T(t,p)*. First order logic.
Derived notions	Tenses: past (*P*), always past (*H*), future(*F*), always future(*G*). Modalities: necessity (□), possibility (◊).

However, Prior also rejected the idea that time is made up of objectively existing instants. Formally this understanding of time can be expressed in terms of the third or fourth grade of tense-logical involvement. According to Prior the crucial idea of the third grade “consists in treating the instant-variables *a, b, c,* etc. as representing propositions.” [21: 124]

Such instant-propositions describe the world uniquely at any given instant, and are for this reason also called world-state propositions. Like Prior we shall use *a, b, c ...* as instant-propositions instead of t_1, t_2, ... In fact, Prior assumed that such propositions *are* what ought to be meant by ‘instants’:

> A world-state proposition in the tense-logical sense is simply an index of an instant; indeed, I would like to say that it is an instant, in the only sense in which ‘instants’ are not highly fictitious entities. [19: 188-189]

The traditional distinction between the description of the content and the indication of time for an event is thereby dissolved. This means that ordinary propositional logic in insufficient here. With his formulation of the third grade, Prior actually became the first logician to explore the formalism of hybrid languages which are now known in so-called hybrid logic (see http://www.hylo.net).

The axioms of the logical language, which embodies the third grade of tense logical involvement, are in the first place the following axioms of a basic tempo-modal logic:

(A1) p, where p is a tautology of the propositional calculus

(A2) $G(p \supset q) \supset (Gp \supset Gq)$

(A3) $H(p \supset q) \supset (Hp \supset Hq)$

(A4) $p \supset GPp$

(A5) $p \supset HFp$

(□1) $(p \supset q) \supset (\Box p \supset \Box q)$

(□2) $\Box p \supset p$

(□3) $\Box p \supset \Box\Box p$

(□G) $\Box p \supset Gp$

(□H) $\Box p \supset Hp$

In addition, in order to deal with the instant propositions involved in the hybrid language of the third grade Prior added the following axioms:

(I1) $\exists a: a$

(I2) $\sim\Box\sim a$

(I3) $\Box(a \supset p) \vee \Box(a \supset \sim p)$

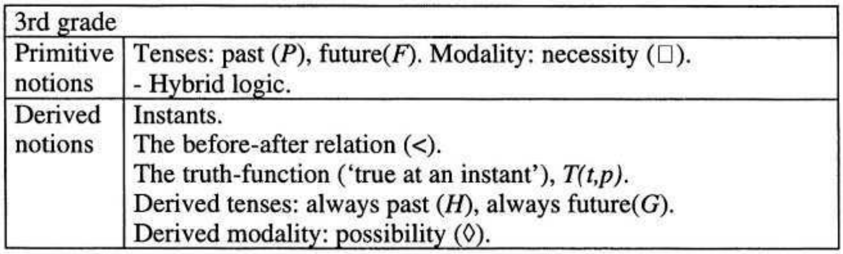
(BF) $\Box(\forall a: \phi(a)) \equiv \forall a: \Box(\phi(a))$

where a stands for an instant proposition, whereas p stands for any proposition. Given these axioms and some standard rules of reference Prior was able to prove a number of interesting theses for this hybrid language corresponding to the third grade. He argued that within this system *T(a,p)* as well as the before-after-relation can be defined in terms of the tenses and a primitive necessity-operator □. Formally, this can be done in the following way:

$$T(a,p) \equiv_{def} \Box(a \supset p)$$
$$a<b \equiv_{def} \Box(b \supset Pa)$$

Prior demonstrated convincingly, that *T* and < defined in this way fulfill everything which holds for the corresponding notions in the first grade. (See [21: 124 ff.]) For instance, Prior proved that the formulae (DF) and (DP) hold with instant propositions instead of temporal instants.

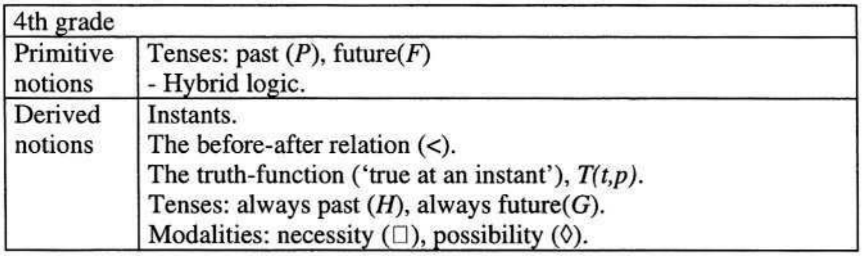
In the third grade all of temporal logic can be developed from the purely ‘modal notions’ of past, future, and necessity. The corresponding time ontology can be presented in the following manner:



3rd grade	
Primitive notions	Tenses: past (*P*), future(*F*). Modality: necessity (□). - Hybrid logic.
Derived notions	Instants. The before-after relation (<). The truth-function (‘true at an instant’), *T(t,p)*. Derived tenses: always past (*H*), always future(*G*). Derived modality: possibility (◊).

In some logics the truth-function involves not only propositions and instants, but also chronicles (i.e. ‘totally ordered subsets of instants’) cf. [31: 211]). In these logics the truth-functions becomes *T(t,c,p),* where *t* is an instant belonging to the chronicle *c*. *T(t,c,p)* can be read ‘*p* is true at *t* on *c*’. However, this complication is not needed in the third grade, where the instant propositions also include the information corresponding to the chronicle.

The fourth grade may be seen as a continuation of the third grade, from which all definitions are carried over. However, according to the view corresponding to the fourth grade a tense logical definition of the necessity-operator can be established, if the future operator is understood as corresponding to ‘possible future’ and not some sort of ‘factual future’ (i.e. in this case ‘future’ has to be take in the Peircean sense cf. [31: 220]). This means that in the fourth grade the only primitive operators are the two tense logical ones: *P* and *F*. Within the hybrid logic all other notions can be defined from these two tense operators. This means that the corresponding time ontology can be presented in the following way:



4th grade	
Primitive notions	Tenses: past (*P*), future(*F*) - Hybrid logic.
Derived notions	Instants. The before-after relation (<). The truth-function (‘true at an instant’), *T(t,p)*. Tenses: always past (*H*), always future(*G*). Modalities: necessity (□), possibility (◊).

Prior himself favoured this fourth grade. It appears that his reasons for wanting to reduce modality to tenses were mainly metaphysical, since it has to do with his rejection of the concept of the (one) true (but still unknown) future.

In the systems corresponding to the third and the fourth grade of tense-logical involvement it is assumed that it is possible to quantify over instant propositions. If Quine's so-called 'criterion of ontological commitment' is accepted, it follows that instants (conceived as instant propositions) actually exist in reality. However, Prior did not accept this conclusion. In fact, Prior rejected Quine's criterion. According to his idea of existence we do not have to accept the existence of instants just because we are prepared to bind them by quantifiers. Existence is rather a question of "what variables we take seriously as individual variables in a first-order theory, i.e. as subjects of predicates rather than as *assertibilia* which may be qualified by modalities" [21: 220]. When viewed in this way, we can conclude that instants do not exist in reality. As Prior put it, "they don't exist; rather, they are or are not the case". [21: 220]

3 The Role of Durations in Time Ontology

The notion of a 'duration' is obviously important, when it comes to the study of temporal ontology. Several logicians have tried to formulate a logic of duration. In fact, already the medieval logician John Buridan (ca. 1295-1358) regarded the present as a duration and not as a point in time. Indeed, he made a number of important contributions to the development durational logic (see [31: 43 ff.].

The first modern logician to formulate a calculus in this field was A.G. Walker [29], who considered a structure $(S,<)$, where S is a non-empty set of periods. The '$a<b$'-relation is to be considered as 'strict' in the sense that no overlap between a and b is permitted, and the ordering is supposed to be irreflexive, asymmetrical, and transitive. In addition he considered the notion of overlap, which can be defined as:

$$a \mid b \equiv_{def} \sim(a<b \vee b<a).$$

Walker formulated an axiomatic system using the following two axioms:

(W1) $a \mid a$
(W2) $(a<b \wedge b \mid c \wedge c<d) \supset a<d$

Using a set-theoretic method base Walker demonstrated that it is possible to define instants in terms of durations. For this reason it may be reasonable to view a temporal instant as such a 'secondary' construct from the logic of durations. In the 1950s C. Lejewski worked out a so-called 'chronology' according to which objects are conceived as corresponding to durations. This work should actually be seen as an extension of the **Leśniewskian** calculus, which appears to have been the first formal ontology. [25: 18 ff.]

In 1972 Charles Hamblin [11] independently also put forth a theory of the logic of durations. He achieved his results using a different technique involving the relation:

$$a \text{ meets } b \equiv_{def} a<b \wedge \sim(\exists c\colon a<c \wedge c<b))$$

A decade later, similar considerations have been put forward within artificial intelligence research, notably by James Allen and Patrick Hayes [3], [4], [5], [6], who has analysed further details of durational logic. Allen and Hayes have shown that two arbitrary durations (in linear time) can be related in exactly 13 ways.

It has been argued in [31: 304 ff.] that Walker's and Hamblin's theories are equivalent when seen from an ontological point of view. They all support the claim that just as durations (temporal interval) set-theoretically can be constructed from an instant-logic, it is possible to do the opposite i.e. to construct instants mathematically from durations. In fact, all the durational theories put forth so far appear to give rise to the same ontological model. Given the set of instants and the before-after relation we may define chronicles as linear ordered subsets of the total set of total set of instants.

The theories formulated by Walker, Hamblin, and Allen can all be said to be B-theoretical. It is however also possible to take an A-theoretical approach to durational logic. This can in fact be done in two different ways. Already John Buridan suggested these two alternative ideas for the construction of the logic of tenses. Firstly, the tenses, past and future, can be taken absolutely, in the sense that no part of the present time is said to be past or future. Secondly, tenses can be taken in the relative sense, according to which "the earlier part of the present time is called past with respect to the later, and the later part is called future with respect to the earlier." [7: 175]

In addition, Buridan pointed out that if some thing is moving now, then there is a part of the present during which it is moving, and hence, it is moving in some part of the present, which is earlier than some other part of the present. Therefore, if the thing is moving, then it was moving (if the past is taken in the relative sense). For this reason, the Aristotelian sophism must be conceded if the past is understood relatively. In a modern formal language Buridan's idea of a relative past can be stated in the following way:

$$T(I_n,P_{rel}A) \equiv_{def} \forall I': included(I',I_n) \supset (\exists I'': I''<I' \wedge T(I'',A))$$

whereas the absolute past can be defined as

$$T(I,P_{abs}p) \equiv_{def} \exists I_1: I< I_1 \wedge T(I_1,p)$$

In modern durational logic a similar distinction has been introduced by Röper and others (see [31: 312 ff]). It turns out, that given that time can be viewed as a structure like the ordered set of real numbers and that durations are just open intervals, it can be shown that $P_{rel} \equiv G_{abs}P_{abs}$. There are of course a lot of details to be worked out here. But in the cases in which this equivalence holds, we obviously do not need to take both sets of tenses as primitives since the relative tenses can be derived from the absolute tenses.

It will be possible to combine the distinction between durations and instants with the above discussion about Prior four grades. If we assume Prior's third grade and that durations should be derived from instants we get an ontology, which can be outlined as indicated in the schema below:

3rd grade extended with durations	
Primitive notions	Tenses: past (*P*), future (*F*). Modality: necessity ($\Box$). - Hybrid logic.
Derived notions	Instants. Chronicles. The before-after relation (<). A truth-function: *T(t,p)* Derived tenses: always past (*H*), always future (*G*). Derived modality: possibility ($\Diamond$).
Secondary derived notions.	Durations. Durational absolute tenses: P_{abs}, H_{abs}, F_{abs}, G_{abs} Durational relative tenses: P_{rel}, H_{rel}, F_{rel}, G_{rel}

On the other hand, if we assume Prior's third grade and that instants should be derived from durations things become much more complicated. But although there are a number of open questions regarding the precise procedures, it is very likely that it will be possible to develop an ontology based on durational tenses and hybrid logic.

4 Analysis of Some Current Time Ontologies

In light of the growing number of available ontologies, and the emerging need to collect information from different ontological descriptions, it becomes increasingly important to evaluate ontologies. It is sometimes said that the better ontology is one, which fulfills its purpose. There is obviously some truth in that, but if the purpose includes interoperability or the ability to merge with other ontologies, this question becomes much more complex. In the process of designing ontologies, the ontologist must on one hand take into account competing or comparable efforts and on the other hand pay special attention to the main purpose of the ontology. The Cyc ontology, for instance, is concerned with encompassing common knowledge whereas the DAML Time ontology deals with how to handle web documents. Clearly, the purpose of the ontology is reflected in the choice of words and concepts as well as in matters like expressive power and granularity.

The question of how to evaluate and compare ontologies must extend beyond matters of aligning lists of concepts, and also take into account the deeper lying intention and metaphysical positions behind the respective ontologies. These questions are broad and complicated, not least because top-level ontologies by nature deal with many different domains. In order to describe and compare different ontological approaches, a larger framework is needed. Top-level ontologies are likely to diverge in many other areas than that of temporal notions. In fact, we believe that a similar effort may be appropriate in several other dimensions than the one under consideration here. However, considering the fundamental status of time, this seems an appropriate place to start. In order to analyze temporal notions in ontologies, we suggest using Prior's four grades. Once the framework has been properly established, the procedure of comparing ontologies simply consists in looking for the relevant primitives from which other notions are defined, and map them to one of the grades. It turns out that such an analysis at the surface level is quite simple, and it is likely that it can be automated, but also that slight differences in terminology and style of representation may complicate such analyses tremendously.

In order to compare some influential time ontologies we shall in the following examine their primitive and derived notions. To our knowledge, most ontologies that deals with temporal notions have *instants* as well as *intervals* as primitives, and as such, they are only slightly different. The DAML Time ontology [12] refers to this as *instants* and *intervals,* and the Suggested Upper Merged Ontology (SUMO) [13] uses *time-points* and *time-intervals.* In OpenCyc [17] TimeInterval is seen as a direct generalization of TimePoint, indicating that TimePoints are defined in terms of intervals, but in effect TimePoints are treated as primitive notions.

The before-after relation is also common to most time ontologies although it is expressed in different ways. In SUMO an *earlier* relation is defined over *intervals* and a *before* relation is defined over *time points* (instants). DAML Time uses only a *before* relation from which all other temporal relations are defined, and OpenCyc has two primitive temporal predicates *(after* and *simultaneousWith)* for defining an order between temporal elements. No before-relation is formally defined in OpenCyc, but the documentation clearly shows how the semantics of *'before'* is distributed over several complex temporal predicates. This choice in ordering corresponds to McTaggart's B-series.

Corresponding to the truth-function *T(a,p),* most ontologies have predicates that link events to times, such as the (holdsIn TEMP-THING FORMULA) of OpenCyc which says that a formula is true at every moment in the extent of TemporalThing i.e. TimeInterval or TimePoint. Ontologies that satisfy these conditions qualify as first grade ontologies. The crucial distinction between first and second grade is the notion of *Now* and the consequences thereof.

In DAML Time this predicament is partially avoided by considering concepts like *'now'* and *'today'* as deictic times, relative to some document or utterance. The documentation briefly mentions the possibility of defining *past* and*future* in terms of this deictic 'now' and the before relation. [12: section 7]. A similar but more explicit solution is used by the TERQAS group [22] associated with the TimeML effort [2]. In identifying temporal attributes of events for Question Answering systems, the TERQAS group distinguishes between absolute and relative time according to the document date [1: section 4]. Likewise, SUMO has no formal definition of *'now'* but the provided mapping between SUMO and WordNet points to 'now' as simply a time point. In consequence hereof, *'past'* and *'future'* does not share ontological status with *time-intervals,* but are defined as functions that map a TimePosition to the TimeInterval which it meets and which begins / ends at PositiveInfinity / NegativeInfinity. In his treatment of processes, John Sowa follows a similar line of reasoning, but in addition, he explicitly defines *'past'* by means of the indexical *'Now'* and the B-like relation *'succession'* [28: 207]. But where he in [27] was primarily concerned with representing grammatical structures [28] goes one step further in accounting for the special status of *now.*

Common to these approaches is that *'now'* is treated as an arbitrary indexical defined in terms of the before-after relation. A more developed notion of now is found in OpenCyc. This ontology does, as the only top level ontology that we are aware of, have the notion of *'Now'* as a primitive, and should therefore be considered a time ontology of the second grade. Surprisingly, OpenCyc has no definitions of *'past'* and *'future'* as anything other than grammatical constructions, but as we have seen, the tempo-logical notions of *past* and *future* can be derived in both the first and second grade.

> #$Now is a special #$TimePoint which denotes the current moment from the perspective of the instantiation of #$CycTheCollection that is currently being run (i.e. #$Cyc). [...] Thus the referent of #$Now does not vary with the #$Microtheory in which one asks (#$indexicalReferent #$Now ?X). Instead, the referent of #$Now varies from moment to moment down to the resolution of #$Cyc's central processing unit. [17]

Prior strongly suggested that in building a temporal ontology we should take the durational existence of things or objects into account. He pointed out that talk about events »is really at bottom talk about things, and that what looks like talk about changes in events is really just slightly more complicated talk about changes in things« [21: 16]. Recently, Karl Erich Wolff and Wendsomde Yameogo [30] have introduced a conceptual analysis involving objects and their so-called 'life tracks'. This approach mainly corresponds to Prior's first grade. Something similar can be said about Pavel Kocura's approach [14] with the addition that this work also involves a durational logic (e.g. reasoning about time intervals).

In 1993 Bernard Moulin [16] has demonstrated how temporal information in discourses can be modelled using the notion of a time coordinate system. In a very clear manner he has shown how the idea of a time axis, i.e. a set of 'time points' and total order relation, <, can be used in the definitions of time intervals, temporal objects and situations. Moulin has also shown how relations like 'during' and even rather complicated tenses can be defined in his model. His approach turns out to be a rather elaborated version of Prior's first grade extended with temporal objects and time intervals (defined from 'time points').

After having reviewed a number of contemporary and influential time ontologies, we have not found any that goes beyond the second grade of tense-logical involvement. None of the ontologies relate time and modality, and only DAML Time silently supports (allows for) branching time models, which certainly gives rise to an interesting challenge.

5 Towards a Tensed Ontology

There is no reason to believe that the struggle between A-theorists and B-theories will ever be settled. This means that we will probably in all foreseeable future have to deal with different time ontologies based on different philosophical positions. It is, however, worth noting that the logic based on Prior's third grade, which take instants as well as durations into account, seems to be the most general language, we have considered in this study. In fact, all concepts from all the time ontologies considered can be translated into such a logical language. This means that a logical language based on the third grade and extended to include durations as well as instants will qualify as a language in terms of which time ontologies may be compared. In particular, such a language will be fit for Prior's ideas of a tensed ontology. None of the other grades can provide a similar platform for comparison between ontologies. There are, however, a number of problems, which should be solved in order to carry out the full Priorean program.

The most common relation in theories of formal ontology is that of a sub-category corresponding to statements of the form ‘x is a y’. Prior argued, however, that a satisfactory ontological system should also take tensed variations of such statements into consideration. This means that we should study statements like ‘x has been a y” and ‘x is a future y’ in an ontological context (see [19: 162 ff].

Things become rather complicated, since Prior promotes the idea that new entities come into being as time passes. Some perfectly sensible statements now, would be non-statable in the past. This means for instance, that we can now say things about the past events, which could not be stated at all when the events took place. This logic of statability is incorporated in Prior’s system Q, according to which the following theses hold:

$$F(n)(\forall x{:}\phi(x)) \supset \forall x{:}F(n)\phi(x)$$
$$P(n)(\exists x{:}\phi(x)) \supset \exists x{:}P(n)\phi(x)$$
$$\forall x{:}P(n)\phi(x) \supset P(n)(\forall x{:}\phi(x))$$

whereas ‘the mirror-images’:

$$P(n)(\forall x{:}\phi(x)) \supset \forall x{:}P(n)\phi(x)$$
$$F(n)(\exists x{:}\phi(x)) \supset \exists x{:}F(n)\phi(x)$$
$$\forall x{:}F(n)\phi(x) \supset F(n)(\forall x{:}\phi(x))$$

do not hold. The reason for this asymmetry is that “the values of bound variables may receive additions but no deletions as time passes” [19: 172]. A full logic corresponding to Prior’s idea of a tensed ontology will have to integrate Prior’s third grade of tense-logical involvement and his Q-system. To the best of our knowledge nobody has tried to formulate a formal ontology corresponding to his idea of a growing conceptual framework. There is obviously a lot to do in order to carry out the full Priorean program about tensed ontology.

References

1. *Annotation Guidelines for Relation Detection and Characterization (RDC) Version 3.5 - 4.22.2002:* http://www.cs.brandeis.edu/~jamesp/arda/time/documentation/RDC-Guidelines-V3.5.doc.
2. *TimeML,* http://www.cs.brandeis.edu/~jamesp/arda/time/.
3. Allen, J.F., *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM, 1983: p. vol. 26 823-843.
4. Allen, J.F., *Towards a General Theory of Action and Time.* Artificial Intelligence 23, 1984.
5. Allen, J.F. and P. Hayes, *A Common-Sense Theory of Time.* Proc. of the Ninth Int. Joint Conf. on Artificial Intelligence, 1985: p. 528-531.
6. Allen, J.F. and P. Hayes, *Moments and Points in an Interval-based Temporal Logic.* Comput. Intell., 1989. **5**: p. 225-238.
7. Buridan, J., *Sophisms on Meaning and Truth.* 1966, New York.
8. Chernyakov, A., *The Ontology of Time. Being and Time in the Philosophies of Aristotle, Husserl and Heidegger.* 2002: Kluwer.

9. Copeland, J., ed. *Logic and Reality: Essays on the Legacy ofArthur Prior.* 1996, Oxford University Press.
10. Gruber, T.R., *A Translation Approach to Portable Ontologies.* Knowledge Acquisition, 1993. **5**(2): p. 199-220.
11. Hamblin, C.L., *Instants and Intervals,* in *The Study of Time,* J.T. Fraser, F.C. Haber, and G.H. Müller, Editors. 1972, Springer-Verlag: Berlin. p. 324-331.
12. Hobbs, J.R., et al., *A DAML Ontology ofTime.* 2002: http://www.cs.rochester.edu/~ferguson/daml/daml-time-nov2002.txt.
13. IEEE, *Suggested Upper Merged Ontology.* 2003, http://ontology.teknowledge.com.
14. Kocura, P., *Representing Temporal Ontology in Conceptual Graphs,* in *Conceptual Structures for Knowledge Creation and Communication,* A.d. Moor, W. Lex, and B. Ganter, Editors. 2003, Springer Verlag. p. 174-187.
15. McTaggart, J.E., *TheUnreality ofTime.* Mind, 1908: p. 457-474.
16. Moulin, B., *The representation oflinguistic information in an approach usedfor modelling temporal knowledge in discourses,* in *ICCS 1993 LNAI 699,* G.W. Mineau, B. Moulin, and J. Sowa, Editors. 1993. p. 182-204.
17. OpenCyc, *Time and Dates.* 2002, http://www.cyc.com/cycdoc/vocab/time-vocab.html.
18. Prigogine, I., *From Being to Becoming, Time and Complexity in the Physical Sciences.* 1980, San Francisco: W. H. Freeman & Co.
19. Prior, A.N., *Past, Present and Future.* 1967, Oxford: Clarendon Press.
20. Prior, A.N., *The Notion of the Present,* in *The Study of Time,* J.T. Fraser, F.C. Haber, and G.H. Müller, Editors. 1972, Springer.
21. Prior, A.N., *Papers on Time and Tense,* ed. P. Hasle, et al. 2003: Oxford University Press.
22. Pustejovsky, J., *TERQAS: Time and Event Recognition for Question Answering Systems.* 2002: http://www.cs.brandeis.edu/~jamesp/arda/time/terqas/index.html.
23. Quine, W.V.O., *On What There Is.* 1953: Reprinted in From a Logical Point of View, New York: Harper & Row.
24. Rescher, N. and A. Urquhart, *Temporal Logic.* 1971: Springer.
25. Simons, P., *A Study in Ontology.* 1987: Clarendon Press, Oxford.
26. Smith, B., *Philosophical Ontology,* in *Guide to the Philosophy of Computing and Information,* L. Floridi, Editor. 2003, Oxford: Blackwell. p. 155-166.
27. Sowa, J., *Conceptual Structures: Information Processing in Mind and Machine.* 1984: Addison-Wesley.
28. Sowa, J., *Knowledge Representation.* 2000: Brooks Cole Publishing Co.
29. Walker, A.G., *Durées et instants.* La Revue Scientifique, 1947(No. 3266): p. 131 ff.
30. Wolff, K.E. and W. Yameogo, *Time Dimension, Objects, and Life Tracks. A Conceptual Analysis,* in *Conceptual Structures for Knowledge Creation and Communication,* A.d. Moor, W. Lex, and B. Ganter, Editors. 2003, Springer Verlag. p. 188-200.
31. Øhrstrøm, P. and P. Hasle, *Temporal Logic - From Ancient Ideas to Artificial Intelligence.* 1995: Kluwer Academic Publishers.

This page intentionally left blank

Author Index