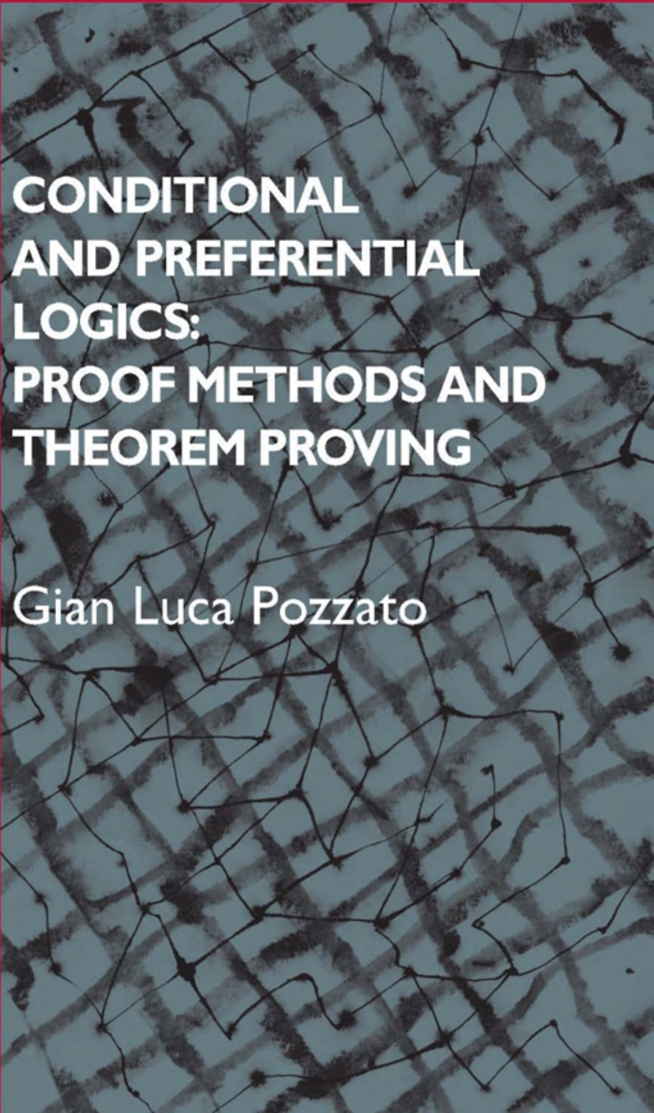
Frontiers
in
Artificial
Intelligence
and
Applications



CONDITIONAL AND PREFERENTIAL LOGICS: PROOF METHODS AND THEOREM PROVING

Gian Luca Pozzato

IOS
Press

CONDITIONAL AND PREFERENTIAL LOGICS: PROOF METHODS AND THEOREM PROVING

Frontiers in Artificial Intelligence and Applications

Volume 208

Published in the subseries

Dissertations in Artificial Intelligence

Under the Editorship of the ECCAI Dissertation Board

Recently published in this series

Vol. 207. A. Bifet, Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams
Vol. 206. T. Welzer Družovec et al. (Eds.), Information Modelling and Knowledge Bases XXI
Vol. 205. G. Governatori (Eds.), Legal Knowledge and Information Systems – JURIX 2009: The Twenty-Second Annual Conference
Vol. 204. B. Apolloni, S. Bassis and C.F. Morabito (Eds.), Neural Nets WIRN09 – Proceedings of the 19th Italian Workshop on Neural Nets
Vol. 203. M. Džbor, Design Problems, Frames and Innovative Solutions
Vol. 202. S. Sandri, M. Sànchez-Marrè and U. Cortés (Eds.), Artificial Intelligence Research and Development – Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence
Vol. 201. J.E. Agudo et al. (Eds.), Techniques and Applications for Mobile Commerce – Proceedings of TAMoCo 2009
Vol. 200. V. Dimitrova et al. (Eds.), Artificial Intelligence in Education – Building Learning Systems that Care: From Knowledge Representation to Affective Modelling
Vol. 199. H. Fujita and V. Mařík (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the Eighth SoMeT_09
Vol. 198. R. Ferrario and A. Oltramari (Eds.), Formal Ontologies Meet Industry
Vol. 197. R. Hoekstra, Ontology Representation – Design Patterns and Ontologies that Make Sense
Vol. 196. F. Masulli et al. (Eds.), Computational Intelligence and Bioengineering – Essays in Memory of Antonina Starita
Vol. 195. A. Boer, Legal Theory, Sources of Law and the Semantic Web
Vol. 194. A. Petcu, A Class of Algorithms for Distributed Constraint Optimization
Vol. 197. R. Hoekstra, Ontology Representation – Design Patterns and Ontologies that Make Sense
Vol. 196. F. Masulli et al. (Eds.), Computational Intelligence and Bioengineering – Essays in Memory of Antonina Starita

ISSN 0922-6389

Conditional and Preferential Logics: Proof Methods and Theorem Proving

Gian Luca Pozzato

Department of Computer Science, Università degli Studi di Torino, Italy

IOS Press

Amsterdam • Berlin • Tokyo • Washington, DC

ISBN 978-1-60750-094-0 (print)
ISBN 978-1-60750-095-7 (online)
Library of Congress Control Number: 2009943096
doi: 10.3233/978-1-60750-095-7-i

Publisher
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

LEGAL NOTICE
The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

Preface

This volume is focused on proof methods and theorem proving for Conditional and Preferential logics. *Conditional logics* are extensions of classical logic by means of a conditional operator, usually denoted as $\Rightarrow$. Conditional logics have a long history, and recently they have found application in several areas of AI, including belief revision and update, the representation of causal inferences in action planning, the formalization of hypothetical queries in deductive databases. Conditional logics have been also applied in order to formalize nonmonotonic reasoning. The study of the relations between conditional logics and nonmonotonic reasoning has lead to the seminal work by Kraus, Lehmann, and Magidor, who have introduced the so called *KLM framework*. According to this framework, a defeasible knowledge base is represented by a finite set of conditional assertions of the form $A \mid\!\sim B$, whose intuitive reading is "*typically (normally), the A's are B's*" or "*B is a plausible conclusion of A*". The operator $\mid\!\sim$ is nonmonotonic in the sense that $A \mid\!\sim B$ does not imply $A \wedge C \mid\!\sim B$. The logics of the KLM framework, also known as *Preferential logics*, allow to infer new conditional assertion from a given knowledge base.

In spite of their significance, very few deductive mechanisms and theorem provers have been developed for Conditional and Preferential logics. In this work we try to (partially) fill the existing gap by introducing proof methods (sequent and tableau calculi) for these logics, as well as theorem provers obtained by implementing the proposed calculi.

This book contains a revised and updated version of my Ph.D. dissertation, which has been awarded by the Italian Association for Logic Programming (GULP) with the "Marco Cadoli" prize, as one of the distinguished dissertations focused on computational logic discussed between 2007 and 2009. First of all, I want to express my gratitude to the members of GULP, in particular to Gianfranco Rossi, Alberto Pettorossi and Paolo Torroni, who have made this publication possible. I would also like to thank the board of trustees of the "Marco Cadoli" award: Agostino Dovier, Andrea Formisano, Enrico Pontelli and Giorgio Delzanno. Moreover, I am very grateful to the referees of my thesis: Rajeev Goré, Guido Governatori, Daniel Lehmann and Dale Miller, for their helpful comments and interesting suggestions, which have helped me a lot in improving the final version of this work.

I would like to thank all the members of the "Logic Programming and Automated Reasoning Group" of the Department of Computer Science of the University of Turin: Alberto Martelli, Maria Luisa Sapino, Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Claudio Schifanella. I want also to express my gratitude to Pietro Torasso, our Ph.D. Coordinator, for the time he has spent for me, especially during the last months of my Ph.D. program. I would also like to thank my friend Roberto Micalizio: he's a great researcher, and he is the best person I've ever met.

I can't find the right words to thank Laura Giordano and Nicola Olivetti for supporting me through the years of my Ph.D. program, and for dedicating me time and patience. Their enthusiasm and their constant encouragement have been fundamental for the realization of my thesis. I will always be in debt with Laura and Nicola: they are two wonderful persons, and I am so lucky to work with them. I want also to express my gratitude to Valentina Gliozzi for her essential suggestions, for sharing her knowledge with me, and moreover for her patience in tolerating my bad temper when we have worked together.

Finally, I would like to thank my parents, for helping and supporting me since I was born, and my wife Eleonora, for sharing with me every second of her life. And I would like to thank my sweet son Lorenzo, for asking me to play with him, for breaking my heart every time I have to leave home, for making me feel his hero.

November 2009 Gian Luca Pozzato

Alla mia Stellina
e a tutti i piccoli Angeli del cielo

Contents

Chapter 1

Introduction

In this chapter we introduce the contents of this work. We give an essential introduction to conditional and preferential logics, observing that, in spite of their significance, very few deductive mechanisms and theorem provers have been developed for them. In this work we try to (partially) fill the existing gap by introducing proof methods (sequent and tableau calculi) for conditional and preferential logics, as well as theorem provers obtained by implementing the proposed calculi.

1.1 Introduction to Automated Reasoning - Monotonic vs Nonmonotonic Inference Systems

The Artificial Intelligence (AI) is a branch of Computer Science that studies if and how an artificial machine can have a *human behavior*. To diagnose a disease given some observed sympthons, to plan the sequence of activities in order to achieve a goal, to find a solution given a (consistent) set of constraints are some problems whose solution is in the scope of AI.

In order to solve this kind of *intelligent problems*, an intelligent system needs to formalize and automatize the *reasoning mechanisms* of human beings. For instance, people usually assume that all the unknown information at a time are false, or they conclude that, given a hierarchy of concepts, properties of individuals of a class (concept) C_1 are inherited by all the items of a class C_2, the latter being a more specific class of C_1.

An intelligent system works within a specific context, known as its *application domain*. In order to achieve its goals, to offer its services, to solve the problems for what it was developed, an intelligent system, as well as an intelligent agent, represents its domain by using a *specific formalism*. The set of items representing the application domain forms the system's *knowledge base*.

Given an *explicit* knowledge base, an intelligent system needs to *infer* new information, that is to say to make *explicit* some information which are only implicit in the knowledge base. Consider, for instance, the knowledge base K containing the following information:

"*Ph.D. students have to defend a thesis*", "*Elisa is a Ph.D. student*"

It is reasonable that any intelligent agent should also infer that

"*Elisa has to defend a thesis*"

in case this information would be needed in order to achieve some goals.

In the literature, reasoning/inference systems are usually divided into:

- monotonic systems;
- nonmonotonic systems.

We say that a reasoning system is *monotonic* if it is characterized by the following property, called *monotonicity*:

Definition 1.1 (Monotonicity)**.** *Suppose that a given intelligent system is able to infer the information A from its knowledge base K. We denote this fact by $K \vdash A$. The reasoning system is monotonic if, given any new information B, we have that $K \cup \{B\} \vdash A$.*

Intuitively, a monotonic reasoning system does not retract any information previously concluded, even if the knowledge base is enriched by some new information. In the above example, suppose that the following information

"*Most Ph.D. students are very nice persons*"

is added to the agent's knowledge base, and suppose also that our agent implements a monotonic reasoning system. It is quite obvious that the conclusion "*Elisa has to defend a thesis*" is still a plausible conclusion of the resulting knowledge base, then it will be inferred by the agent's reasoning machinery.

Classical logic is characterized by the monotonicity property, i.e. if A is a logical consequence of a set of formulas K, then, given any formula B, A is also a logical consequence of $K \cup \{B\}$. Therefore, all the proof methods developed for classical logic, such as resolution, describe *only* monotonic reasoning systems.

However, in everyday life, human beings often make reasonings of a different nature; indeed, it is usual to *retract an information* previously inferred when *learning a new fact*, for instance because the "old" conclusion is now inconsistent with the updated knowledge. A reasoning system is called *nonmonotonic* if it is not characterized by the monotonicity property; this means that an agent is able to retract an information A previously inferred, i.e. A can no longer be inferred by the reasoning machinery, given a new information B. As an example, consider the following knowledge base:

"*Most cats wash their hair by themselves*", "*Eolo is a cat*"

It is quite obvious that

(1) "*Eolo washes its hair by itself*"

can be considered a plausible conclusion. However, when learning the following facts:

"*Sphinxs are hairless cats*", "*Eolo is a sphinx*" then (1) is no longer an information that a reasonable reasoning system should infer. An intelligent system should be able to retract it, since the knowledge base has been enriched by the facts that Eolo is a sphinx and, since sphinx are hairless cats, it will never wash its hair. Notice that, when learning the new information, the intelligent agent *does not* have to change its mind about the facts that most cats wash their hair, and that Eolo is a cat; however, it has to change its mind about a conclusion previously inferred as a consequence of an *incomplete knowledge base*.

From what described above, it follows that an intelligent system must be able to perform nonmonotonic reasonings. Indeed, there are lots of situations and problems in which an intelligent system has to reason in presence of incomplete, or inconsistent, or dynamic information. For instance, in *planning* (reasoning about actions) the formalization of a dynamic domain requires to solve the well known *frame problem*, i.e. the problem of modelling the persistency of those facts of the world that are not affected by the execution of actions. In knowledge and data bases, it is usual to assume that an information not explicitly stated to be true is considered to be false.

Many systems exhibiting a nonmonotonic behavior have been studied in the literature; *negation as failure* [Cla78], *Circumscription* [McC80], Reiter's *default logic* [Rei80] are only some examples. Each of these systems deserves an independent interest, however it is not clear that any one of them really captures the whole general properties of nonmonotonic reasoning. In [Var88] a number of researchers expressed their disappointment with respect to the existing systems of nonmonotonic reasoning, and suggested that no purely logical analysis could be satisfactory to this aim.

A first step toward the formalization of nonmonotonic reasoning systems consists in the application of *conditional and preferential logics*.

1.2 Conditional Logics

Conditional logics extend classical logic by means of a *conditional operator*, typically denoted by $\Rightarrow$. Their recent history starts with Lewis in the 70s [Lew73], who adopted conditional logics in order to formalize a kind of hypothetical reasoning with sentences like "*if A were the case then B*", that cannot be captured by the implication with its classical meaning. Moreover, they have been used to formalize sentences of the form "*A implies B*", where the antecedent A is always false, known in the literature as *counterfactual conditional sentences*. In recent years, conditional logics have found application in several areas of AI, including belief revision and update, the representation of causal inferences in action planning, the formalization of hypothetical queries in deductive databases. We give a broader discussion on applications of conditional logics in Section 2.4.

Similarly to modal logics, the semantics of conditional logics can be defined in terms of possible world structures. In this respect, conditional logics can be seen as a generalization of modal logics (or a type of multi-modal logic) where the conditional operator is a sort of modality indexed by a formula of the same language.

The two most popular semantics for conditional logics are the so-called *sphere semantics* [Lew73] and the *selection function semantics* [Nut80]. Both are possible-world semantics, but are based on different (though related) algebraic notions. In this work we adopt the selection function semantics, which is more general and considerably simpler than the sphere semantics.

Since we adopt the selection function semantics, CK is the fundamental system; it has the same role as the system K (from which it derives its name) in modal logic: CK-valid formulas are exactly those ones that are valid in every selection function model.

Conditional logics have been also applied in order to formalize nonmonotonic reasoning. First, Delgrande [Del87] proposed a conditional logic for prototypical reasoning, in which a conditional formula $A \Rightarrow B$ was interpreted as "*the A's have typically the property B*". In Delgrande's conditional logic, the knowledge base about cats introduced in the previous example can be represented as follows:

$$\forall x(\mathit{Sphinx(x)} \rightarrow \mathit{Cat(x)})$$
$$\forall x(\mathit{Sphinx(x)} \rightarrow \neg\ \mathit{WashHair(x)})$$
$$\forall x(\mathit{Cat(x)} \Rightarrow\ \mathit{WashHair(x)})$$

The last sentence states that cats *typically* wash their hair. Observe that when replacing $\Rightarrow$ with the classical implication $\rightarrow$, the above knowledge base is consistent only if there are no sphinx: and what about Eolo?

1.3 Preferential Logics

The study of the relations between conditional logics and nonmonotonic reasoning has gone much further since the seminal work by Kraus, Lehmann, and Magidor [KLM90], who have introduced the so called *KLM framework*. According to this framework, a *defeasible* knowledge base is represented by a finite set of *conditional assertions* of the form $A \mid\!\sim B$, whose intuitive reading is "*typically (normally), the A's are B's*" or "*B is a plausible conclusion of A*". The operator $\mid\!\sim$ is nonmonotonic in the sense that $A \mid\!\sim B$ does not imply $A \wedge C \mid\!\sim B$.
As an example, consider the following defeasible knowledge base:

$$\mathit{home\_theater\_device} \mid\!\sim \neg \mathit{battery}$$
$$\mathit{remote\_controller} \mid\!\sim \mathit{home\_theater\_device}$$
$$\mathit{remote\_controller} \mid\!\sim \mathit{battery}$$

This knowledge base represents that, typically, devices for home theater (i.e. LCD TV color, DVD recorders, audio equipment, and so on) do not need batteries, whereas remote controllers are devices for home theater requiring batteries. In some of the logics belonging to the KLM framework, from the above knowledge base, one can infer, for instance, that $\mathit{home\_theater\_device} \mid\!\sim \neg \mathit{remote\_controller}$ (typical home theater devices are not remote controllers) or that $\mathit{home\_theater\_device} \wedge \mathit{remote\_controller} \mid\!\sim \mathit{battery}$ (giving preference to more specific information). In none of the logics one can conclude unacceptable information such as $\mathit{remote\_controller} \mid\!\sim \neg \mathit{battery}$.

The set of axioms and inference rules adopted determines the four different logics of the framework, namely (from the strongest to the weakest): *Rational logic* **R**, *Preferential logic* **P**, *Loop-Cumulative logic* **CL**, and *Cumulative logic* **C**. The relationship between conditional logics and KLM logics has been widely investigated; it turns out that all forms of inference studied in KLM framework are particular cases of well-known conditional axioms [CL92]. In this respect the KLM language is just a fragment of conditional logics.

Kraus, Lehmann and Magidor proposed the KLM framework in order to describe a set of postulates that *any concrete nonmonotonic reasoning system should satisfy.* Furthermore, they have shown that the corresponding axiom systems are sound and complete with respect to a semantics based on a *preference relation* (see an introductive discussion here below). The logics of KLM framework are also known as *preferential logics* or *logics of plausible reasoning.*

As mentioned, from a semantic point of view, to each logic (**C**, **CL**, **P**, **R**) corresponds one kind of models, namely, possible-world structures equipped with a preference relation among worlds or states. More precisely, for **P** we have models with a preference relation (an irreflexive and transitive relation) on worlds. For the stronger **R** the preference relation is further assumed to be *modular.* For the weaker logic **CL**, the transitive and irreflexive preference relation is defined on *states*, where a state can be identified, intuitively, with a set of worlds. In the weakest case of **C**, the preference relation is on states, as for **CL**, but it is no longer assumed to be transitive. In all cases, the meaning of a conditional assertion $A \mid\sim B$ is that B holds in the *most preferred* worlds/states where A holds.

In KLM framework the operator "$\mid\sim$" is considered as a meta-language operator, rather than as a connective in the object language. However, it has been readily observed that KLM systems **P** and **R** coincide to a large extent with the flat (i.e. unnested) fragments of well-known conditional logics, once we interpret the operator "$\mid\sim$" as a binary connective.

In recent years, Halpern and Friedman [FH01] have shown that the axiom systems of some KLM logics are complete with respect to a wide spectrum of different semantics (e.g. possibilistic structures and κ-rankings), proposed in order to formalize some forms of nonmonotonic reasoning. This can be explained by the fact that all these models are examples of *plausibility structures*, and the truth in them is captured by the axioms of some preferential logics. These results, and their extensions to the first order setting [FHK00] are the source of a renewed interest in KLM framework.

1.4 State of the Art

In spite of their significance, very few proof systems have been proposed for conditional and preferential logics. As a consequence, even less theorem provers have been implemented. Here we recall the state of the art for what concerns proof methods for both conditional and preferential logics. A *proof method* is a machinery able to verify if an information A can be entailed by a knowledge base K, as well as to check whether K is (in)consistent; in other words, a proof method is a *calculus* that is used to automatize an inference mechanism. It is curious enough that, in spite of their application to nonmonotonic reasoning, proof methods and theorem provers for conditional and preferential logics have been partially neglected.

1.4.1 Conditional Logics

As mentioned, very few proof systems have been developed for conditional logics. We just mention [Lam92, DG90, CdC95, AGR02, Gen92, dS83, GGOS03]. One possible reason of the underdevelopment of proof-methods for conditional logics is the lack of a universally accepted semantics for them. This is in sharp contrast to modal and temporal logics which have a consolidated semantics based on a standard kind of Kripke structures.

Most of the works in the literature have concentrated on extensions of CK. The investigation of proof systems has been addressed to conditional logics based on both of the two most popular semantics: Crocco, Fariñas, Artosi, Governatori, and Rotolo have studied proof systems for conditional logics based on the selection function semantics, whereas De Swart, Gent, Lamarre, Groeneboer, and Delgrande have studied proof systems for logics with sphere semantics.

- Crocco and Fariñas [CdC95] present sequent calculi for some conditional logics including CK, CEM, CO and other axioms. Their calculi comprise two levels of sequents: principal sequents with $\vdash_P$, corresponding to the basic deduction relation, and auxiliary sequents with $\vdash_a$, corresponding to the conditional operator: thus the constituents of $\Gamma \vdash_P \Delta$ are sequents of the form $X \vdash_a Y$, where X, Y are sets of formulas. The bridge between auxiliary sequents and conditional formulas is given by the *descending rule*:

  $$\frac{X_1 \vdash_a B_1, \ldots, X_{n-1} \vdash_a B_{n-1} \vdash_P X_n \vdash_a B_n}{A_1 \Rightarrow B_1, \ldots, A_{n-1} \Rightarrow B_{n-1} \vdash_P A_n \Rightarrow B_n}$$

  where the $A_i = \bigwedge_i X_i$. These systems provide an interesting proof-theoretical interpretation of the conditional operator in terms of structural rules (eg. reduction and augmentation rules). It is not clear if these calculi can be used to obtain a decision procedure for the respective logics.

- Artosi, Governatori, and Rotolo [AGR02] develop labelled tableau for the conditional logic CU that corresponds to the cumulative logic **C** of KLM framework. Formulas are labelled by path of worlds containing also variable worlds. Their tableau method is defined primarily for a conditional language without nested conditionals; however, the authors discuss a possible extension to a non restricted conditional language. They do not use a specific rule to deal with equivalent antecedents of conditionals. They use instead a sophisticated unification procedure to propagate positive conditionals. The unification process itself checks the equivalence of the antecedents. Their tableau system is based on KE and thus it contains a cut rule, called PB, whose application can be restricted in an analytic way.

- De Swart [dS83] and Gent [Gen92] give sequent/tableaux calculi for the strong conditional logics VC and VCS. Their proof systems are based on the entrenchment connective $\leq$, from which the conditional operator can be defined. Their systems are analytic and comprise an infinite set of rules $\leq F(n, m)$, with a uniform pattern, to decompose each sequent with m negative and n positive entrenchment formulas.

- Lamarre [Lam92] presents tableaux systems for the conditional logics V, VN, VC, and VW. Lamarre's method is a consistency-checking procedure which tries to build a system of spheres falsifying the input formulas. The method makes use of a subroutine to compute the *core*, that is defined as the set of formulas characterizing the minimal sphere. The computation of the core needs in turn the consistency checking procedure. Thus there is a mutual recursive definition between the procedure for checking consistency and the procedure to compute the core.

- Groeneboer and Delgrande [DG90] have developed a tableau method for the conditional logic VN which is based on the translation of this logic into the modal logic S4.3.

We also mention the following works on conditional logics with a *preferential* semantics, which is related to the sphere semantics.

- [GGOS03] and [GGOS05] have defined a labelled tableau calculus for the logic CE and some of its extensions. The flat fragment of CE corresponds to the preferential logic **P** of the KLM framework. The tableau calculus makes use of pseudo-formulas, that are modalities in a hybrid language indexed on worlds. In that paper it is shown how to obtain a decision procedure for that logic by performing a kind of loop checking.
- Finally, complexity results for some conditional logics have been obtained by Friedman and Halpern [FH94]. Their results are based on a semantic analysis, by an argument about the size of possible countermodels. They do not give an explicit decision procedure for the logics studied. As mentioned, they consider conditional logics with the preferential semantics; since in this work we adopt the selection function semantics, the systems they consider are either stronger or not comparable with ours[1]. Most of the logics they consider turn out to be PSPACE complete.

1.4.2 KLM Logics

As mentioned above, even if KLM was born as an inferential approach to nonmonotonic reasoning, curiously enough, there has not been much investigation on deductive mechanisms for these logics. The state of the art can be summarized as follows:

- Lehmann and Magidor [LM92] have proved that validity in **P** is **coNP**-complete. Their decision procedure for **P** is more a theoretical tool than a practical algorithm, as it requires to guess sets of indexes and propositional evaluations. They have also provided another procedure for **P** that exploits its reduction to **R**. However, the reduction of **P** to **R** breaks down if boolean combinations of conditionals are allowed, indeed it is exactly when such combinations are allowed that the difference between **P** and **R** arises.
- As described in the previous section, a tableau proof procedure for **C** has been given in [AGR02]. In this work authors also show how to extend the system to Loop-Cumulative logic **CL** and discuss some ways to extend it to the other logics.
- As mentioned in the previous section, in [GGOS03] and [GGOS05] some labelled tableaux calculi have been defined for the conditional logic **CE** and its main extensions, including axiom **CV**. The flat fragment (i.e. without nested conditionals) of **CE**(+**CV**) corresponds to **P** (and to **R**, respectively). These calculi however need a fairly complicated loop-checking mechanism to ensure

[1] Among the others, they consider the semantic condition of centering, which is known as MP, but they do not consider strong centering CS nor CEM studied in this work.

termination. It is not clear if they match the complexity bounds and if they can be adapted in a simple way to **CL** and to **C**.

- Finally, decidability of **P** and **R** has also been obtained by interpreting them into standard modal logics, as it is done by Boutilier [Bou94]. However, his mapping is not very direct and natural. We discuss it in detail in Section 1.5.2.
- To the best of our knowledge, for **CL** no decision procedure and complexity bound was known before the present work.

1.5 Motivation and Contribution of This Work

We have discussed above that, in spite of their significance, very few proof systems and theorem provers have been developed for conditional and preferential logics. With this work, we intend to (partially) fill this gap, providing analytic calculi for some *standard conditional logics* based on the selection function semantics and for *all the four KLM systems*. In both cases, we restrict our concern to propositional logics.

In the following sections, we discuss the content of this work and how it is organized in detail.

1.5.1 Proof Methods for Conditional Logics

In Chapter 4, we present a sequent calculus for CK and for some of its standard extensions, namely CK+{ID, MP, CS, CEM} including most of the combinations of them. To the best of our knowledge, the presented calculi are the first ones for these logics. Our calculi make use of labels, following the line of [Vig00], [Gab96], and [AG98]. Two types of formulas are involved in the rules of the calculi: world formulas of the form $x : A$, representing that A holds at world x, and transition formulas of the form $x \xrightarrow{A} y$, representing that $y \in f(x, A)$, where f is the selection function. The rules manipulate both kinds of formulas.

We are able to give cut-free calculi for CK and all its extensions in the set {ID, MP, CS, CEM}, except those including *both* CEM and MP. The completeness of the calculi is an immediate consequence of the admissibility of cut.

We show that one can derive a decision procedure from the cut-free calculi. Whereas the decidability of these systems was already proved by Nute (by a finite-model property argument) [Nut80], our calculi give the first *constructive* proof of decidability. As usual, we obtain a terminating proof search mechanism by controlling the backward application of some critical rules. By estimating the size of the finite derivations of a given sequent, we also obtain a polynomial space complexity bound for these logics.

We can also obtain a tighter complexity bound for the logics CK{+ID}, as they satisfy a kind of *disjunction property*.

Our calculi can be the starting point to develop goal-oriented proof procedures, according to the paradigm of Uniform Proofs by Miller and others [MNPS91, GO00]. Calculi of these kind are suitable for logic programming applications. As a preliminary result we present a goal-directed calculus for a fragment of CK and its extensions with ID and MP, where the “clauses” are a sort of conditional Harrop formulas.

1.5.2 Proof Methods for Preferential Logics

In Chapter 5, we introduce tableau procedures for all KLM logics. Starting with the preferential logic **P**, we provide proof methods based on tableaux calculi for all the logics of KLM framework. We also provide complexity bounds for these logics. Our approach is based on a novel interpretation of **P** into modal logics. As a difference with previous approaches (e.g. Crocco and Lamarre [CL92] and Boutillier [Bou94]), that take S4 as the modal counterpart of **P**, we consider here Gödel-Löb modal logic of provability G. Our tableau method provides a sort of run-time translation of **P** into modal logic G.

The idea is simply to interpret the preference relation as an accessibility relation: a conditional $A \mid\!\sim B$ holds in a model if B is true in all A-worlds w, i.e. worlds in which A holds, that are minimal. An A-world is minimal if all smaller worlds are not A-worlds. The relation with modal logic G is motivated by the fact that we assume, following KLM, the so-called *smoothness condition*, which is related to the well-known *limit assumption*. This condition ensures indeed that minimal A-worlds exist, by preventing an infinitely descending chain of worlds. This condition is therefore ensured by the finite-chain condition on the accessibility relation (as in modal logic G). Therefore, our interpretation of conditionals is different from the one proposed by Boutilier, who rejects the smoothness condition and then gives a less natural (and more complicated) interpretation of **P** into modal logic S4.

However, we do not give a formal translation of **P** into G, we appeal to the correspondence as far as it is needed to derive the tableau rules for **P**. For deductive purposes, we believe that our approach is more direct, intuitive, and efficient than translating **P** into G and then using a calculus for G.

We are able to extend our approach to the cases of **CL** and **C** by using a second modality which takes care of states. Regarding **CL** we show that we can map **CL**-models into **P**-models with an additional modality. The very fact that one can interpret **CL** into **P** by means of an additional modality does not seem to be previously known and might be of independent interest. In both cases, **P** and **CL**, we can define a decision procedure and obtain also a complexity bound for these logics, namely that they are both **coNP**-complete. In case of **CL** this bound is new, to the best of our knowledge.

We treat **C** in a similar way: we can establish a mapping between Cumulative models and a kind of bi-modal models. However, because of the lack of transitivity, the target modal logic is no longer G. The reason is that the *smoothness condition* (for any formula A, if a state satisfies A either it is minimal or it admits a smaller minimal state satisfying A) can no longer be identified with the finite-chain condition of G. As a matter of fact, the smoothness condition for **C** cannot be identified with any property of the accessibility relation, as it involves unavoidably the evaluation of formulas in worlds. We can still derive a tableau calculus based on our semantic mapping. But we pay a price: as a difference with **P** and **CL** the calculus for **C** requires a sort of (analytic) cut rule to account for the smoothness condition. This calculus gives nonetheless a decision procedure for **C**.

Finally, we consider the case of the strongest logic **R**; as for the other weaker systems, our approach is based on an interpretation of **R** into an extension of modal logic G, including modularity of the preference relation (previous approaches [CL92, Bou94] take S4.3 as the modal counterpart of **R**). As a difference with the tableau calculi introduced for **P**, **CL**, and **C**, here we develop a *labelled* tableau system, which seems to be the most natural approach in order to capture the modularity of the preference relation. The calculus defines a systematic procedure which allows the satisfiability problem for **R** to be decided in nondeterministic polynomial time, in accordance with the known complexity results for this logic.

1.5.3 Theorem Provers

In Chapter 6 we describe two implementations of the sequent and tableau calculi introduced in Chapters 4 and 5, resulting in two theorem provers for conditional and preferential logics.

In detail, we present CondLean, a theorem prover for conditional logics, and KLMLean, a theorem prover for KLM logics. To the best of our knowledge, these are the first theorem provers for the respective logics.

Both CondLean and KLMLean are SICStus Prolog implementations of the sequent/tableaux calculi, inspired by the so called "*lean*" methodology introduced by Beckert and Posegga, in which every clause of a predicate `prove` implements an axiom or rule of the calculus and the proof search is provided for free by the mere depth-first search mechanism of Prolog, without any ad hoc mechanism [BP95, Fit98].

CondLean and KLMLean also comprise a graphical user interface written in Java.

Furthermore, we present a very simple SICStus Prolog implementation of a goal-directed proof procedure for the basic normal conditional logic CK and its extensions with ID and MP. This goal-directed theorem prover is called GOALD$\mathcal{U}$CK.

Chapter 2

Conditional Logics

This chapter introduces conditional logics. Conditional logics have a long history, and recently they have found several applications in different areas of artificial intelligence and knowledge representation.

In this chapter we list all the main systems of conditional logics known in the literature, and we describe their axiom systems. Moreover, we describe the two most popular semantics for conditional logics: the *selection function* semantics and the *sphere semantics*. Both of them are possible-world semantics, but are based on different (though related) algebraic notions. Furthermore, we mention a third proposal of semantics for conditional logics, introduced in [Bur81] and called *preference-based* semantics.

This chapter also highlights the main applications of conditional logics in artificial intelligence and knowledge representation, such as nonmonotonic reasoning, hypothetical reasoning, belief revision, and also diagnosis.

2.1 Introduction

The recent history of the conditional logics starts with the work by Lewis [Lew73, Nut80, Che75, Sta68], who proposed these logics in order to formalize a kind of hypothetical reasoning (if A were the case then B), that cannot be captured by classical logic with material implication. Moreover, they have been used to formalize *counterfactual sentences*, i.e. conditionals of the form "if A were the case then B would be the case", where A is false. If we interpret the *if...then* in the above sentence as a classical implication, we obtain that all counterfactuals are trivially true. Nonetheless one may want to reason about counterfactuals sentences and hence be capable of distinguishing among true and false ones. For instance, consider the following counterfactual:

"*After the World Cup final match, if the police had controlled my alcohol level, then it would have retired my driving licence*"

An intelligent system could need to evaluate the above counterfactual, even as false. On the one hand, the agent could evaluate the counterfactual as true, therefore it could suggest the driver to have more control in his future drinks; on the other hand, the agent could evaluate the counterfactual as false, thus considering acceptable the driver's behavior. If the implication were interpreted with its classical meaning, then the (not so much...) intelligent agent would evaluate the above counterfactual as true, then having no arguments to suggest the driver to refrain from drinking before driving his car. For a broader discussion about counterfactuals we refer to [CM99].

Recently, conditional logics have found application in several fields of artificial intelligence, such as knowledge representation, nonmonotonic reasoning, deductive databases, and natural language semantics. In knowledge representation, conditional logics have been used to reason about prototypical properties [Gin86], to model database update [Gra98], belief revision [Bou94, GGO02], causal inference in action planning [Sch99] and diagnosis [Gin86, Obe01]. Moreover conditional logics can provide an axiomatic foundation of nonmonotonic reasoning [KLM90], as it turns out that all forms of inferences studied in the framework of nonmonotonic logics are particular cases of conditional axioms [CL92]; we will discuss this relationship in Section 3.3.

Let us describe conditional logics in a more technical detail.

Conditional logics are extensions of classical logic obtained by adding the *conditional operator*, a binary connective typically denoted by $\Rightarrow$. In the literature, the conditional operator is also denoted by $>$ or $\supset$.

In this work, we restrict our concern to *propositional* conditional logics. A propositional conditional language $\mathcal{L}$ contains the following items:

- a set of propositional variables ATM;
- the symbol of *false* $\bot$;
- a set of connectives $\rightarrow$, $\Rightarrow$.

The usual connectives $\top$, $\wedge$, $\vee$ and $\neg$ can be defined in terms of $\bot$ and $\rightarrow$.
We define formulas of $\mathcal{L}$ as follows:

- $\bot$ and the propositional variables of ATM are *atomic formulas*;
- if A and B are formulas, $A \rightarrow B$ and $A \Rightarrow B$ are *complex formulas.*

For instance, given $P, Q, R \in ATM$, the formulas $(P \Rightarrow Q) \Rightarrow \bot$ and $(P \rightarrow R) \rightarrow (Q \Rightarrow \bot)$ are formulas of $\mathcal{L}$.

2.2 Axiomatizations of Conditional Logics

In this section we present the axiom systems of the most important conditional logics introduced in the literature. Here below is a list of rules and axioms characterizing conditional logics, together with definitions of all well known conditional logics. For a broader discussion, see [Nut80] and [Nut84].

Rules

- (RCEA) $$\frac{A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$$

- (RCK) $\dfrac{(A_1 \wedge \cdots \wedge A_n) \rightarrow B}{(C \Rightarrow A_1 \wedge \cdots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)}$
- (RCEC) $\dfrac{A \leftrightarrow B}{(C \Rightarrow A) \leftrightarrow (C \Rightarrow B)}$
- (RCE) $\dfrac{A \rightarrow B}{A \Rightarrow B}$

Axioms

- (AC) $(A \Rightarrow B) \wedge (A \Rightarrow C) \rightarrow (A \wedge C \Rightarrow B)$
- (CM) $(A \Rightarrow (B \wedge C)) \rightarrow ((A \Rightarrow B) \wedge (A \Rightarrow C))$
- (CC) $((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \Rightarrow (B \wedge C))$
- (ID) $A \Rightarrow A$
- (MP) $(A \Rightarrow B) \rightarrow (A \rightarrow B)$
- (MOD) $(\neg A \Rightarrow A) \rightarrow (B \Rightarrow A)$
- (CSO) $((A \Rightarrow B) \wedge (B \Rightarrow A)) \rightarrow ((A \Rightarrow C) \leftrightarrow (B \Rightarrow C))$
- (RT) $((A \wedge B) \Rightarrow C) \rightarrow ((A \Rightarrow B) \rightarrow (A \Rightarrow C))$
- (CV) $((A \Rightarrow B) \wedge \neg(A \Rightarrow \neg C)) \rightarrow ((A \wedge C) \Rightarrow B)$
- (CA) $((A \Rightarrow B) \wedge (C \Rightarrow B)) \rightarrow ((A \wedge C) \Rightarrow B)$
- (CS) $(A \wedge B) \rightarrow (A \Rightarrow B)$
- (CEM) $(A \Rightarrow B) \vee (A \Rightarrow \neg B)$
- (SDA) $((A \vee B) \Rightarrow C) \rightarrow ((A \Rightarrow C) \wedge (B \Rightarrow C))$
- (CUT) $((A \Rightarrow B) \wedge ((A \wedge B) \Rightarrow C)) \rightarrow (A \Rightarrow C)$

We define conditional logics in each case as the smallest conditional logic closed under certain rules and containing certain axiom schemata. The most important conditional logics are the following ones (we adopt the notation used in [Nut80]):

- Ce = ⟨ RCEC ⟩
- CE = ⟨ RCEC, RCEA ⟩
- Cm = ⟨ RCEC; CM ⟩
- CM = ⟨ RCEC, RCEA; CM ⟩
- Cr = ⟨ RCEC; CM, CC ⟩
- CR = ⟨ RCEC, RCEA; CM, CC ⟩
- Ck = ⟨ RCEC, RCK ⟩
- G = ⟨ RCEC, RCEA, RCE ⟩
- CK = ⟨ RCEC, RCEA, RCK ⟩
- Ck+ID = ⟨ RCEC, RCK; ID ⟩
- CK+ID = ⟨ RCEC, RCEA, RCK; ID ⟩
- Ck+MP = ⟨ RCEC, RCK; MP ⟩

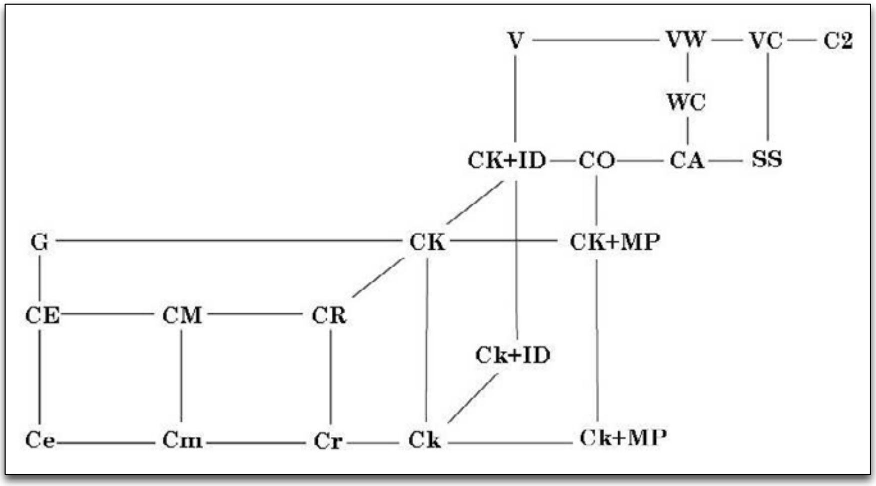


Figure 2.1: Conditional logics according to their containment [Nut80].

- CK+MP = ⟨ RCEC, RCEA, RCK; MP ⟩
- CO = ⟨ RCEC, RCK; ID, MP, MOD, CSO ⟩
- CA = ⟨ RCEC, RCK; ID, MP, MOD, CSO, CA ⟩
- WC = ⟨ RCEC, RCK; ID, MP, MOD, CSO, CA, RT ⟩
- V = ⟨ RCEC, RCK; ID, MOD, CSO, CV ⟩
- VW= ⟨ RCEC, RCK; ID, MOD, CSO, CV, MP ⟩
- VC = ⟨ RCEC, RCK; ID, MOD, CSO, CV, MP, CS ⟩
- SS = ⟨ RCEC, RCK; ID, MP, MOD, CSO, CA, CS ⟩
- C2 = ⟨ RCEC, RCK; ID, MOD, CSO, CV, MP, CEM ⟩

In [Nut80] it is shown that all of these logics are consistent. Figure 2.1 is taken from [Nut80] and shows the relationships between conditional logics. One logic contains one another if the former is above or to the right of the latter in the diagram. All the containment relations indicated are proper.

2.3 The Semantics of Conditional Logics

Similarly to modal logics, the semantics of conditional logics can be defined in terms of possible world structures. In this respect, conditional logics can be seen as a generalization of modal logics (or a type of multi-modal logic) where the conditional operator is a sort of modality indexed by a formula of the same language.

The two most popular semantics for conditional logics are the so-called *sphere semantics* [Lew73] and the *selection function semantics* [Sta68, Nut80]. Both are possible-world semantics, but are based on different (though related) algebraic notions. The relation between sphere and selection function semantics has been widely investigated by Grahne [Gra98], who has shown their equivalence for some systems.

In this section we describe the two approaches in detail. The selection function semantics is more general and considerably simpler than the sphere semantics. Other semantics have been proposed for conditional logics, such as *relational models*, *neighborhood models* and *extensional models*. In addition to the sphere and the selection function semantics, here we recall the so called *preference-based* semantics, introduced in [Bur81]. For a complete picture of all the other semantics for conditional logics, we refer to Chapter 3 in [Nut80].

2.3.1 The Selection Function Semantics

The selection function semantics has been introduced by Stalnaker [Sta68] (see [Nut80] for a survey). This semantics is based on the presence of a *selection function* f. Truth values are assigned to formulas depending on a world; intuitively, the selection function f selects, for a world w and a formula A, the set of worlds $f(w, A)$ which are "most-similar to w" or "closer to w" given the information A. We restrict our concern to *normal* conditional logics, in which the function f depends on the set of worlds satisfying A rather than on A itself, so that $f(w, A) = f(w, A')$ whenever A and A' are true in the same worlds (normality condition). This condition is the semantic counterpart of the rule (RCEA). A conditional sentence $A \Rightarrow B$ is true in w whenever B is true in every world selected by f for A and w. It is the normality condition which essentially marks the difference between conditional logics on the one hand, and multimodal logic, on the other hand (where one might well have a family of $\Box$ indexed by formulas). We believe that it is the very condition of normality what makes difficult to develop proof systems for conditional logics with the selection function semantics.

Considering the selection function semantics, CK is the fundamental system; it has the same role as the system K (from which it derives its name) in modal logic: CK-valid formulas are exactly those that are valid in every selection function model.

Let us describe the selection function semantics in detail.

Definition 2.1 (Selection function semantics)**.** *A model is a triple*

$$\mathcal{M} = \langle \mathcal{W}, f, [\,]\rangle$$

where:

- $\mathcal{W}$ *is a non empty set of items called* worlds*;*
- f *is the so-called* selection function *and has the following type:*

$$f \colon \mathcal{W} \times 2^{\mathcal{W}} \longrightarrow 2^{\mathcal{W}}$$

- $[\,]$ *is the* evaluation function*, which assigns to an atom $P \in ATM$ the set of worlds where P is true, and is extended to the other formulas as follows:*

  $\star$ $[\bot] = \emptyset$

  $\star$ $[A \rightarrow B]$=($\mathcal{W}$ - $[A]$) $\cup$ $[B]$

  $\star$ $[A \Rightarrow B]$=$\{w \in \mathcal{W} \mid f(w,\, [A]) \subseteq [B]\}$

A formula A is valid in a model $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$, denoted with $\mathcal{M} \models A$, iff $[A] = \mathcal{W}$. A formula A is valid, denoted with $\models A$, iff it is valid in every model.

Observe that we have defined f taking $[A]$ rather than A (i.e. $f(w,[A])$ rather than $f(w,A)$) as an argument; this is equivalent to define f on formulas, i.e. $f(w,A)$ but imposing that if $[A]$=$[A^{'}]$ in the model, then $f(w,A)$=$f(w,A^{'})$. This condition is called *normality*.

The semantics above characterizes the *basic conditional system*, called CK. Other conditional systems are obtained by assuming further properties on the selection function; here is the list of well known standard extensions of the basic system CK:

Name	**Axiom**	**Model condition**
ID	$A \Rightarrow A$	$f(w,[A]) \subseteq [A]$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	$w \in [A] \rightarrow w \in f(w,[A])$
CS	$(A \wedge B) \rightarrow (A \Rightarrow B)$	$w \in [A] \rightarrow f(w,[A]) \subseteq \{w\}$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w,[A]) \mid \leq 1$
AC	$(A \Rightarrow B) \wedge (A \Rightarrow C) \rightarrow (A \wedge C \Rightarrow B)$	$f(w,[A]) \subseteq [B] \rightarrow f(w,[A \wedge B]) \subseteq f(w,[A])$
RT	$(A \wedge B \Rightarrow C) \rightarrow ((A \Rightarrow B) \rightarrow (A \Rightarrow C))$	$f(w,[A]) \subseteq [B] \rightarrow f(w,[A]) \subseteq f(w,[A \wedge B])$
CV	$(A \Rightarrow B) \wedge \neg(A \Rightarrow \neg C) \rightarrow (A \wedge C \Rightarrow B)$	$(f(w,[A]) \subseteq [B]$ and $f(w,[A]) \cap [C] \neq \emptyset) \rightarrow f(w,[A \wedge C]) \subseteq [B]$
CA	$(A \Rightarrow B) \wedge (C \Rightarrow B) \rightarrow (A \vee C \Rightarrow B)$	$f(w,[A \vee B]) \subseteq f(w,[A]) \cup f(w,[B])$

The axiomatizations presented in the previous section are complete with respect to the semantics. Given a system S, to denote that A is a theorem in (follows from) the axiomatization S we write $\vdash_S A$.

Theorem 2.2 (Completeness of axiomatization, [Nut80])**.** *If a formula A is valid in S then it is a theorem in the respective axiomatization, i.e. $\vdash_S A$.*

2.3.2 The Sphere Semantics

The sphere semantics has been introduced by Lewis in [Lew73]. The basic idea under this kind of semantics can be illustrated as follows: a system of spheres around a world w is used to represent the *relative similarity* of other worlds to w.

Definition 2.3 (Sphere semantics)**.** *A system of spheres model is a pair*

$$\mathcal{M} = \langle \mathcal{W}, \$ \rangle$$

such that:

- $\mathcal{W}$ *is a non-empty set of items called worlds;*
- $\$$ *assigns to each world* $w \in \mathcal{W}$ *a nested set* $\$_w$ *of subsets of* $\mathcal{W}$*, in the sense that for each* $s_1, s_2 \in \$_w$ *either* $s_1 \subset s_2$ *or* $s_2 \subset s_1$*, which is closed under unions and finite intersections.*

We write $[A]$ *to denote the set of worlds where* A *holds, i.e.* $[A] = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A\}$*, where* $\mathcal{M}, w \models A$ *is defined as follows:*

- *if* A *is a propositional formula, then* $\mathcal{M}, w \models A$ *is defined as in classical logic;*
- $\mathcal{M}, w \models A \Rightarrow B$ *iff* $\bigcup \$_w \cap [A] = \emptyset$ *or there exists an* $S \in \$_w$ *such that* $\emptyset \neq S \cap [A] \subseteq [B]$*.*

A formula A *is valid in a model* $\mathcal{M} = \langle \mathcal{W}, \$ \rangle$*, denoted by* $\mathcal{M} \models A$*, iff* $[A] = \mathcal{W}$*. A formula* A *is valid, denoted by* $\models A$*, iff it is valid in every model.*

As mentioned above, a system of spheres around a world tells us the similarity of other worlds to that world. If $S \in \$_w$, $w' \in S$ and $w'' \notin S$, then w' is more similar to w than w''.

The conditional logic **V** is the basic system of logics based on sphere semantics; indeed, in [Lew73] it is shown that **V** is determined by the class of all sphere semantics models.

2.3.3 The Preference-Based Semantics

The Preference-Based semantics has been introduced in [Bur81]. The idea of this semantics is that every world w has associated a preference relation $\leq_w$ (in general a preorder relation) on the class of worlds. A conditional $A \Rightarrow B$ is true at w if B is true in all A-worlds that are *minimal* with respect to the relation $\leq_w$. The preference-based semantics has been taken as the "official" semantics of conditional logics by Friedman and Halpern [FH94] with, however, one important difference from the setting presented here. In fact, similarly to the semantics of KLM logics presented in the next chapter, the preference-based semantics described here embodies the *limit assumption* (corresponding to the *smoothness condition* in preferential semantics): every non-empty set of worlds has a minimal element with respect to each preorder relation $\leq_x$. This property is not assumed in [FH94].

Definition 2.4 (Preference-based semantics)**.** *A model* $\mathcal{M}$ *has the form*

$$\mathcal{M} = \langle \mathcal{W}, \{\leq_x\}_{x \in \mathcal{W}}, I \rangle$$

where

- $\mathcal{W}$ *is a non-empty set of items called worlds;*
- I *is a function* $\mathcal{W} \rightarrow 2^{ATM}$*;*
- $\{\leq_x\}_{x \in \mathcal{W}}$ *is a family of relations on* $\mathcal{W}$ *for each element* $x \in \mathcal{W}$*.*

For $S \subseteq \mathcal{W}$ *we define*

$$Min_x(S) = \{a \in S \mid \forall b \in S (b \leq_x a \rightarrow a \leq_x b)\}.$$

We say that $Min_x(S)$ is the set of $\leq_x$-minimal elements of S.
We assume the following facts, for every $x \in \mathcal{W}$:

1. *$\leq_x$ is a reflexive and transitive relation on $\mathcal{W}$;*

2. *for every non-empty $S \subseteq \mathcal{W}$, $Min_x(S) \neq \emptyset$.*

We define the truth conditions of formulas with respect to worlds in a model $\mathcal{M}$, denoted by the relation $\mathcal{M}, x \models A$, as follows:

1. *$\mathcal{M}, x \models P$, if $P \in I(x) \cap ATM$*

2. *$\mathcal{M}, x \not\models \bot$*

3. *$\mathcal{M}, x \models A \Rightarrow B$ if for all $y \in Min_x(A)$, $\mathcal{M}, y \models B$, where $Min_x(A)$ stands for $Min_x(\{y \in \mathcal{W} \mid \mathcal{M}, y \models A\})$*

We say that A is valid in $\mathcal{M}$ if $\mathcal{M}, x \models A$ for every $x \in \mathcal{W}$. We say that A is valid if it is valid in every model.

We also define the *strict relation* $y <_x z$ iff $y \leq_x z \wedge \neg(z \leq_x y)$. Observe that $Min_x(S) = \{a \in S \mid \neg\exists b \in S.\ b <_x a\}$.

The above is the semantics of the basic system **CE**. We obtain extensions of this basic system by imposing specific restrictions on the preference relations; some examples follow:

(CV) $y \leq_x z \vee z \leq_x y$ (connectedness)

(CS) $y \leq_x x \rightarrow y = x$ (strong centering)

(MP) $y \leq_x x \rightarrow x \leq_x y$ (weak centering)

(CEM) $y \neq z \rightarrow y <_x z \vee z <_x y$ (conditional excluded middle)

2.4 Applications of Conditional Logics

As mentioned at the beginning of this chapter, in the last years, there has been a considerable amount of work on applications of conditional logics to various areas of artificial intelligence and knowledge representation such as nonmonotonic reasoning, hypothetical reasoning, belief revision, and even diagnosis.

The application of conditional logics in the realm of nonmonotonic reasoning was first investigated by Delgrande [Del87] who proposed a conditional logic for prototypical reasoning; the understanding of a conditional $A \Rightarrow B$ in his logic is "the A's have typically the property B". For instance, one could have:

$\forall x(\textit{Penguin(x)} \rightarrow \textit{Bird(x)})$
$\forall x(\textit{Penguin(x)} \rightarrow \neg\textit{Fly(x)})$
$\forall x(\textit{Bird(x)} \Rightarrow \textit{Fly(x)})$

The last sentence states that birds typically fly. As already observed, replacing $\Rightarrow$ with the classical implication $\rightarrow$, the above knowledge base is consistent only if there are no penguins.

Conditional logics have also been used to formalize knowledge update and revision. For instance, Grahne presents a conditional logic (a variant of Lewis' VCU) to formalize knowledge-update as defined by Katsuno and Mendelzon [Gra98]. More recently in [GGO05] and [GGO02], it has been shown a tight correspondence between AGM revision systems and a specific conditional logic, called BCR. The connection between revision/update and conditional logics can be intuitively explained in terms of the so-called Ramsey Test (RT): the idea is that $A \Rightarrow B$ "holds" in a knowledge base K if and only if B "holds" in the knowledge base K revised/updated with A; this can be expressed by

$$\text{(RT)} \quad K \vdash A \Rightarrow B \text{ iff } K \circ A \vdash B,$$

where $\circ$ denotes a revision/update operator. Observe that on the one hand (RT) gives a definition of the conditional $\Rightarrow$ in terms of a belief-change operator; on the other hand, the conditional logic resulting from the (RT) allows one to describe (and to reason on) the effects of revision/update within the language of conditional logics.

Conditional logics have also been used to model hypothetical queries in deductive databases and logic programming; the conditional logic CK+ID is the basis of the logic programming language CondLP defined in [GGM$^+$00]. In that language one can have hypothetical goals of the form (quoting the old Yale's Shooting problem)

$$Load\_gun \Rightarrow (Shoot \Rightarrow Dead)$$

and the idea is that the hypothetical goal succeeds if *Dead* succeeds in the state "revised" first by *Load_gun* and then by *Shoot*[1].

In a related context, conditional logics have been used to model causal inference and reasoning about action execution in planning [Sch99, GS04]. In the causal interpretation the conditional $A \Rightarrow B$ is interpreted as "A causes B"; observe that identity (i.e. $A \Rightarrow A$) is not assumed to hold.

In order to model actions and causations, Judea Pearl has defined a theory of causal reasoning based on the language of *structural equations* [Pea99, Pea00]. In [GP98], Galles and Pearl study the causal interpretation of counterfactual sentences using the structural equation model for causality. They show that causal implication as defined by causal models satisfies all the axioms and rules of Lewis' conditional logic. In particular, their system includes axioms (ID), (MP) and (CS).

Moreover, conditional logics have found some applications in diagnosis, where they can be used to reason counterfactually about the expected functioning of system components in face of the observed faults [Obe01].

[1] The language CondLP comprises a nonmonotonic mechanism of revision to preserve the consistency of a program potentially violated by an hypothetical assumption.

Chapter 3

KLM Logics

In this chapter we present KLM logics, introduced by Kraus, Lehmann, and Magidor [KLM90, LM92] in order to describe the properties that any concrete nonmonotonic reasoning system should satisfy. The study of the relations between conditional logics and nonmonotonic reasoning has gone much further. It turns out that all forms of inference studied in KLM framework are particular cases of well-known conditional axioms [CL92]. In this respect the KLM language is just a fragment of conditional logics.

We introduce the axiom systems of KLM logics, then we describe their semantics based on a standard kind of Kripke structures equipped with a *preference relation*, distinguishing between the four different logics, namely (from the strongest to the weakest): *rational* logic **R**, *preferential* logic **P**, *loop-cumulative* logic **CL**, and *cumulative* logic **C**.

3.1 Introduction to KLM Logics: a Logical Approach to Nonmonotonic Reasoning

3.1.1 Nonmonotonic Reasoning

Intelligent systems are able to reason about a specific domain of interest. As already mentioned in the Introduction, they represent this domain by a *knowledge base* containing *explicit* information about it. Moreover, an intelligent system must be able to *extract* other information by means of suitable inference rules.

We usually refer to *inference* as the mechanism of "extracting" explicit information that was only implicit in the intelligent system's knowledge base.

We define *monotonic* an inference process respecting the *monotonicity* property introduced in Definition 1.1. This property characterizes those systems whose reasoning mechanism is such that adding a new information to the knowledge base will not cause the retraction of any other information previously inferred.

In everyday life, however, human beings often infer sensible conclusions from what they know and, in front of new information, they have to correct or retract previous conclusions, for instance because they are now in contradiction with the updated reality. We call *nonmonotonic* this kind of inference mechanisms. Nonmonotonic inference systems are those *not* respecting the monotonicity property of Definition 1.1. As already observed, an intelligent system should be able to perform some kind of nonmonotonic reasoning. Many systems exhibiting a nonmonotonic behavior have been studied in the literature; the most popular are the *negation as failure* [Cla78], the *Circumscription* [McC80], the modal systems of [MD80, Moo85], Reiter's *default logic* [Rei80], autoepistemic logic [Moo84] and the *inheritance systems* [Tou86]. Each of these systems deserves an independent interest, however it is not clear that any one of them really captures the whole general properties of nonmonotonic reasoning. In [Var88] a number of researchers expressed their disappointment at existing systems of nonmonotonic reasoning, and suggested that no purely logical analysis could be satisfactory to this aim. Some years later, Kraus, Lehmann and Magidor tried to contradict this pessimistic outlook, as described in the next section.

3.1.2 The Solution of Kraus, Lehmann, and Magidor: the KLM Framework

In the early 90s [KLM90] Kraus, Lehmann and Magidor (from now on KLM) proposed a formalization of nonmonotonic reasoning that was early recognized as a landmark. Their work stemmed from two sources: the theory of *nonmonotonic consequence relations* initiated by Gabbay [Gab85] and the *preferential semantics* proposed by Shoham [Sho87b] as a generalization of Circumscription. KLM's works led to a classification of nonmonotonic consequence relations, determining a hierarchy of stronger and stronger systems. The so called *KLM properties* have been widely accepted as the "conservative core" of nonmonotonic reasoning. The role of KLM logics is similar to the role of AGM postulates in Belief Revision [Gar88]: they give a set of postulates for default reasoning that any concrete reasoning mechanism should satisfy.

According to the KLM framework, defeasible knowledge is represented by a (finite) set of nonmonotonic conditionals or assertions of the form

$$A \mid\!\sim B$$

whose reading is *normally (or typically) the A's are B's*. The operator "$\mid\!\sim$" is nonmonotonic, in the sense that $A \mid\!\sim B$ does not imply $A \wedge C \mid\!\sim B$. For instance, a knowledge base K may contain the following set of conditionals:

$$\begin{array}{l} \mathit{adult} \mid\!\sim \mathit{worker} \\ \mathit{adult} \mid\!\sim \mathit{taxpayer} \\ \mathit{student} \mid\!\sim \mathit{adult} \\ \mathit{student} \mid\!\sim \neg \mathit{worker} \\ \mathit{student} \mid\!\sim \neg \mathit{taxpayer} \\ \mathit{retired} \mid\!\sim \mathit{adult} \\ \mathit{retired} \mid\!\sim \neg \mathit{worker} \end{array}$$

whose meaning is that adults typically work, adults typically pay taxes, students are typically adults, but they typically do not work, nor do they pay taxes, and so on. Observe that if $\mid\!\sim$ were interpreted as classical (or intuitionistic) implication, we simply would get $\mathit{student} \mid\!\sim \bot$, $\mathit{retired} \mid\!\sim \bot$, i.e. typically there are not students, nor retired people, thereby obtaining a trivial knowledge base.

One can derive new conditional assertions from the knowledge base by means of a set of inference rules. In the KLM framework, the set of adopted inference rules defines some fundamental types of inference systems, namely, from the weakest to the strongest: Cumulative (**C**), Loop-Cumulative (**CL**), Preferential (**P**) and Rational (**R**) logic. All these systems allow one to infer new assertions from a given knowledge base K without incurring the trivialising conclusions of classical logic: concerning our example, in none of them, one can infer $student \mid\sim worker$ or $retired \mid\sim worker$. In cumulative logics (both **C** and **CL**) one can infer $adult \land student \mid\sim \neg worker$ (giving preference to more specific information), in Preferential logic **P** one can also infer that $adult \mid\sim \neg retired$ (i.e. typical adults are not retired). In the rational case **R**, if one further knows that $\neg(adult \mid\sim \neg married)$, i.e. it is not the case the adults are typically unmarried, one can also infer that $adult \land married \mid\sim worker$.

From a semantic point of view, to each logic (**C**, **CL**, **P**, **R**) there corresponds one kind of models, namely a class of possible-world structures equipped with a preference relation among worlds or states. More precisely, for **P** we have models with a preference relation (an irreflexive and transitive relation) on worlds. For the strongest **R** the preference relation is further assumed to be *modular*. For the weaker logic **CL**, the transitive and irreflexive preference relation is defined on *states*, where a state can be identified, intuitively, with a set of worlds. In the weakest case of **C**, the preference relation is on states, as for **CL**, but it is no longer assumed to be transitive. In all cases, the meaning of a conditional assertion $A \mid\sim B$ is that B holds in the *most preferred* worlds/states where A holds.

In the KLM framework the operator "$\mid\sim$" is considered as a meta-language operator, rather than as a connective in the object language. However, it has been readily observed that KLM systems **P** and **R** coincide to a large extent with the flat (i.e. unnested) fragments of well-known conditional logics, once we interpret the operator "$\mid\sim$" as a binary connective [CL92, Bou94, KS91]. We will analyze in detail the relationship between KLM logics and conditional logics in Section 3.3.

A recent result by Halpern and Friedman [FH01] has shown that preferential and rational logic are quite natural and general systems: surprisingly enough, the axiom system of preferential (likewise of rational) logic is complete with respect to a wide spectrum of semantics, from ranked models, to parametrized probabilistic structures, ϵ-semantics and possibilistic structures. The reason is that all these structures are examples of *plausibility structures* and the truth in them is captured by the axioms of preferential (or rational) logic. These results, and their extensions to the first order setting [FHK00] are the source of a renewed interest in the KLM framework. A considerable amount of research in the area has then concentrated in developing concrete mechanisms for plausible reasoning in accordance with KLM systems (**P** and **R** mostly). These mechanisms are defined by exploiting a variety of models of reasoning under uncertainty (ranked models, belief functions, possibilistic logic, etc. [BDP97, BSS00, Wey03, Pea90, Mak05, Mak03, Bil93]) that provide, as we remarked, alternative semantics to KLM systems. These mechanisms are based on the restriction of the semantics to preferred classes of models of KLM logics; this is also the case of Lehmann's notion of rational closure (not to be confused with the logic **R**). More recent research has also explored the integration of KLM framework with paraconsistent logics [AA00]. Finally, there has been some recent investigation on the relation between KLM systems and decision-theory [DFPP02, DFP03].

REF. $A \mid\!\sim A$

LLE. If $\vdash_{PC} A \leftrightarrow B$, then $\vdash (A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$

RW. If $\vdash_{PC} A \rightarrow B$, then $\vdash (C \mid\!\sim A) \rightarrow (C \mid\!\sim B)$

CM. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \wedge B \mid\!\sim C)$

AND. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \mid\!\sim B \wedge C)$

OR. $((A \mid\!\sim C) \wedge (B \mid\!\sim C)) \rightarrow (A \vee B \mid\!\sim C)$

RM. $((A \mid\!\sim B) \wedge \neg(A \mid\!\sim \neg C)) \rightarrow ((A \wedge C) \mid\!\sim B)$

Figure 3.1: Axioms and rules of KLM logic **R**. We use $\vdash_{PC}$ to denote provability in the propositional calculus, whereas $\vdash$ is used to denote provability in **R**.

3.2 KLM Logics

In this section we present axiomatizations and semantics of the KLM systems. For the sake of exposition, we present the systems in the order from the strongest to the weakest: **R**, **P**, **CL**, and **C**. For a complete picture of KLM systems, we refer to [KLM90, LM92].

The language of KLM logics consists just of conditional assertions $A \mid\!\sim B$. We consider here a richer language allowing boolean combinations of assertions and propositional formulas. Our language $\mathcal{L}$ is defined from:

- a set of propositional variables ATM;
- the boolean connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$;
- the conditional operator $\mid\!\sim$.

We use $A, B, C, ...$ to denote propositional formulas, whereas $F, G, ...$ are used to denote all formulas (including conditionals). The formulas of $\mathcal{L}$ are defined as follows:

- if A is a propositional formula, $A \in \mathcal{L}$;
- if A and B are propositional formulas, $A \mid\!\sim B \in \mathcal{L}$;
- if F is a boolean combination of formulas of $\mathcal{L}$, $F \in \mathcal{L}$.

3.2.1 Rational Logic R

The axiomatization of **R** consists of all axioms and rules of propositional calculus together with the axioms and rules in Figure 3.1.
REF (*reflexivity*) states that A is always a default conclusion of A. LLE (*left logical equivalence*) states that the syntactic form of the antecedent of a conditional formula is irrelevant. RW (*right weakening*) describes a similar property of the consequent. This allows to combine default and logical reasoning [FH01]. CM (*cautious monotonicity*) states that if B and C are two default conclusions of A, then adding one of the two

conclusions to A will not cause the retraction of the other conclusion. AND states that it is possible to combine two default conclusions. OR states that it is allowed to reason by cases: if C is the default conclusion of two premises A and B, then it is also the default conclusion of their disjunction. RM is the rule of *rational monotonicity*, which characterizes the logic $\mathbf{R}$[1]: if $A \mid\!\sim B$ and $\neg(A \mid\!\sim \neg C)$ hold, then one can infer $A \wedge C \mid\!\sim B$. This rule allows a conditional to be inferred from a set of conditionals in absence of other information. More precisely, "it says that an agent should not have to retract any previous defeasible conclusion when learning about a new fact the negation of which was not previously derivable" [LM92].

The semantics of $\mathbf{R}$ is defined by considering possible world structures with a strict partial order $<$, i.e. an irreflexive and transitive relation, that we call *preference relation*. The meaning of $w < w'$ is that w is preferred to w'. The preference relation is also supposed to be *modular*: for all w, w_1 and w_2, if $w_1 < w_2$ then either $w_1 < w$ or $w < w_2$. We have that $A \mid\!\sim B$ holds in a model $\mathcal{M}$ if B holds in all *minimal worlds* (with respect to the relation $<$) where A holds. This definition makes sense provided minimal worlds for A exist whenever there are A-worlds. This is ensured by the *smoothness condition* in the next definition.

Definition 3.1 (Semantics of $\mathbf{R}$, Definition 14 in [LM92]). *A rational model is a triple*

$$\mathcal{M} = \langle \mathcal{W}, <, V \rangle$$

where:

- *$\mathcal{W}$ is a non-empty set of items called worlds;*
- *$<$ is an irreflexive, transitive and modular relation on $\mathcal{W}$;*
- *V is a function $V : \mathcal{W} \longmapsto 2^{ATM}$, which assigns to every world w the set of atoms holding in that world.*

We define the truth conditions for a formula F as follows:

- *If F is a boolean combination of formulas, $\mathcal{M}, w \models F$ is defined as for propositional logic;*
- *Let A be a propositional formula; we define $Min_<(A) = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A$ and $\forall w'$, $w' < w$ implies $\mathcal{M}, w' \not\models A\}$;*
- *$\mathcal{M}, w \models A \mid\!\sim B$ if for all w', if $w' \in Min_<(A)$ then $\mathcal{M}, w' \models B$.*

(smoothness condition). *The relation $<$ satisfies the following condition, called* smoothness*: if $\mathcal{M}, w \models A$, then $w \in Min_<(A)$ or $\exists w' \in Min_<(A)$ such that $w' < w$.*

We say that a formula F is satisfiable if there exists a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ and a world $w \in \mathcal{W}$ such that $\mathcal{M}, w \models F$. A formula F is valid in a model $\mathcal{M}$*, denoted with $\mathcal{M} \models F$, if $\mathcal{M}, w \models F$ for every $w \in \mathcal{W}$. A formula is* valid *if it is valid in every model $\mathcal{M}$.*

Observe that the above definition of rational model extends the one given by KLM to boolean combinations of formulas. Notice also that the truth conditions for conditional formulas are given with respect to single possible worlds for uniformity sake.

[1] As we will see in Section 3.2.2, the axiom system of the weaker logic $\mathbf{P}$ can be obtained from the axioms of $\mathbf{R}$ without RM.

Since the truth value of a conditional only depends on global properties of $\mathcal{M}$, we have that: $\mathcal{M}, w \models A \mid\sim B$ just in case for all worlds w' of the model, it holds that $\mathcal{M}, w' \models A \mid\sim B$, i.e. $\mathcal{M} \models A \mid\sim B$

The smoothness condition together with the transitivity of $<$ implies the following *strong smoothness condition*:

(**strong smoothness condition**). For all A and w, if there is a world w' preferred to w that satisfies A (i.e. if $\exists w' : w' < w$ and $\mathcal{M}, w' \models A$), then there is also a *minimal* such world (i.e. $\exists w'' : w'' \in Min_<(A)$ and $w'' < w$).

This follows immediately: by the smoothness condition, since $\mathcal{M}, w' \models A$, either $w' \in Min_<(A)$ (and the property immediately follows) or $\exists w''$ s.t. $w'' < w'$ and $w'' \in Min_<(A)$; in turn, by transitivity $w'' < w$, hence the property follows.

It is easy to see that in *frames* $F = \langle \mathcal{W}, < \rangle$ where the $<$ is irreflexive and transitive, the smoothness condition entails that $<$ does not have infinite descending chains. Here is a proof: suppose that F contains an infinite descending chain $\sigma = w_0, w_1, \ldots, w_i \ldots$, with $w_{i+1} < w_i$. Let P be an atom and let V be a propositional evaluation such that $P \in V(w)$ if and only if $w \in \sigma$, then the smoothness condition on P is violated by each world in σ, since for each $w_i \in \sigma$, we have $w_i \models P$ but there is not a *minimal* $w' < w_i$ such that $w' \models P$. Observe that this is a property on frames and not on models.

Observe that, by the modularity of $<$, it follows that possible worlds of $\mathcal{W}$ are *clustered* into equivalence classes, each class consisting of worlds that are incomparable to one another; the classes are totally ordered[2]. In other words the property of modularity determines a *ranking* of worlds so that the semantics of **R** can be specified equivalently in terms of *ranked* models [LM92].

By means of the modularity condition on the preference relation, we can also prove the following theorem. We write $A \mid\sim B \in_+ \Gamma$ (resp. $A \mid\sim B \in_- \Gamma$) if $A \mid\sim B$ occurs positively (resp. negatively) in Γ, where positive and negative occurrences are defined in the standard way.

Theorem 3.2 (Small Model Theorem)**.** *For any $\Gamma \subseteq \mathcal{L}$, if Γ is satisfiable in a rational model, then it is satisfiable in a rational model containing at most n worlds, where n is the size of Γ, i.e. the length of the string representing Γ.*

Proof. Let Γ be satisfiable in a rational model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$, i.e. $\mathcal{M}, x_0 \models \Gamma$ for some $x_0 \in \mathcal{W}$. We build the model $\mathcal{M}' = \langle \mathcal{W}', <', V' \rangle$ as follows:

- We build the set of worlds $\mathcal{W}'$ by means of the following procedure:

```
1. $\mathcal{W}' \longleftarrow \{x_0\}$;
2. for each $A_i \mid\sim B_i \in_- \Gamma$ do
     – choose $x_i \in \mathcal{W}$ s.t. $x_i \in Min_<(A_i)$ and $\mathcal{M}, x_i \not\models B_i$;
     – $\mathcal{W}' \longleftarrow \mathcal{W}' \cup \{x_i\}$;
3. for each $A_i \mid\sim B_i \in_+ \Gamma$ do
     if $Min_<(A_i) \neq \emptyset$, and $Min_< A_i \cap \mathcal{W}' = \emptyset$ then
     – choose any $x_i \in Min_<(A_i)$;
```

[2] Notice that the worlds themselves may be incomparable since the relation $<$ is not assumed to be (weakly) connected.

    - $\mathcal{W}' \longleftarrow \mathcal{W}' \cup \{x_i\}$;

- For all $x_i, x_j \in \mathcal{W}'$, we let $x_i <' x_j$ if $x_i < x_j$;
- For all $x_i \in \mathcal{W}'$, we let $V'(x_i) = V(x_i)$.

In order to show that $\mathcal{W}'$ is a rational model satisfying Γ, we can show the following Facts:

Fact 3.3. $|\mathcal{W}'| \leq n$.

Proof of Fact 3.3. The proof immediately follows by construction of $\mathcal{W}'$.

□ *Fact 3.3*

Fact 3.4. *$\mathcal{M}'$ is a rational model, since $<'$ is irreflexive, transitive, modular and satisfies the smoothness condition.*

Proof of Fact 3.4. Irreflexivity, transitivity and modularity of $<'$ obviously follow from the definition of $<'$. The smoothness condition is ensured by the fact that $<'$ does not have infinite descending chains, since $<$ does not have.

□ *Fact 3.4*

Fact 3.5. *For all $x_i \in \mathcal{W}'$, for all propositional formulas A, $\mathcal{M}, x_i \models A$ iff $\mathcal{M}', x_i \models A$.*

Proof of Fact 3.5. By induction on the complexity of A. The proof is easy and left to the reader.

□ *Fact 3.5*

Fact 3.6. *For all $x_i \in \mathcal{M}'$, for all formulas A s.t. A is the antecedent of some conditional occurring in Γ, we have that $x_i \in Min_{<'}(A)$ iff $x_i \in Min_{<}(A)$.*

Proof of Fact 3.6. First, we prove that if $x_i \in Min_{<'}(A)$, then $x_i \in Min_{<}(A)$. Let $x_i \in Min_{<'}(A)$. Suppose that $x_i \notin Min_{<}(A)$. Since A is the antecedent of a conditional in Γ, by construction of $\mathcal{W}'$, $\mathcal{W}'$ contains x_j, $x_j \neq x_i$, s.t. $x_j \in Min_{<}(A)$ in $\mathcal{M}$. Since $\mathcal{M}, x_i \models A$ and $x_j \in Min_{<}(A)$, we have that $x_j < x_i$. By Fact 3.5, $\mathcal{M}', x_j \models A$, and by the definition of $<'$, $x_j <' x_i$, which contradicts the assumption that $x_i \in Min'_{<}(A)$. We conclude that $x_i \in Min_{<}(A)$ in $\mathcal{M}$.

Now we prove that if $x_i \in Min_{<}(A)$, then $x_i \in Min_{<'}(A)$. Let $x_i \in Min_{<}(A)$ in $\mathcal{M}$. Suppose that $x_i \notin Min_{<'}(A)$. Then there is x_j s.t. $\mathcal{M}', x_j \models A$ and $x_j <' x_i$. By Fact 3.5 (since A is a propositional formula), also $\mathcal{M}, x_j \models A$, and by definition of $<'$, $x_j < x_i$, which contradicts the assumption that $x_i \in Min_{<}(A)$. Hence, $x_i \in Min_{<'}(A)$.

□ *Fact 3.6*

Fact 3.7. *For all conditional formulas $(\neg)A \mathrel{|\!\sim} B$ occurring in Γ, if $\mathcal{M}, x_0 \models (\neg)A \mathrel{|\!\sim} B$, then $\mathcal{M}', x_0 \models (\neg)A \mathrel{|\!\sim} B$.*

Proof of Fact 3.7. We distinguish the two cases:

- $\mathcal{M}, x_0 \models \neg(A \mathrel{|\!\sim} B)$: by construction of $\mathcal{W}'$, there is $x_i \in \mathcal{W}'$ s.t. $x_i \in Min_<(A)$ and $\mathcal{M}, x_i \not\models B$. By Facts 3.5 and 3.6, $x_i \in Min_{<'}(A)$ and $\mathcal{M}', x_i \not\models B$, hence $\mathcal{M}', x_0 \models \neg(A \mathrel{|\!\sim} B)$.
- $\mathcal{M}, x_0 \models A \mathrel{|\!\sim} B$: consider any $x_i \in Min_{<'}(A)$, by Fact 3.6 $x_i \in Min_<(A)$, hence $\mathcal{M}, x_i \models B$, and, by Fact 3.5, $\mathcal{M}', x_i \models B$. We conclude that $\mathcal{M}', x_0 \models A \mathrel{|\!\sim} B$.

□ *Fact 3.7*

By the Facts above, we have shown that Γ is satisfiable in a rational model containing at most n worlds, hence the Theorem follows.

■

In our tableau calculus for **R**, that we will introduce in Section 5.5, we need a slightly extended language $\mathcal{L}_R$. $\mathcal{L}_R$ extends $\mathcal{L}$ by formulas of the form $\Box A$, where A is propositional, whose intuitive meaning is that $\Box A$ holds in a world w if A holds in all the worlds preferred to w (i.e. in all w' such that $w' < w$). We extend the notion of rational model to provide an evaluation of boxed formulas as follows:

Definition 3.8 (Truth condition of modality $\Box$)**.** *We define the truth condition of a boxed formula as follows:*

$$\mathcal{M}, w \models \Box A \textit{ if, for every } w' \in \mathcal{W}, \textit{ if } w' < w \textit{ then } \mathcal{M}, w' \models A$$

From definition of $Min_<(A)$ in Definition 3.1 above, and Definition 3.8, it follows that for any formula A, $w \in Min_<(A)$ iff $\mathcal{M}, w \models A \land \Box\neg A$.

Notice that by the strong smoothness condition, it holds that if $\mathcal{M}, w \not\models \Box\neg A$, then $\exists w' < w$ such that $\mathcal{M}, w' \models A \land \Box\neg A$. If we regard the relation $<$ as the inverse of the accessibility relation R (thus xRy if $y < x$), it immediately follows that the strong smoothness condition is an instance of the property G restricted to A propositional ($\neg\Box\neg A \to \Diamond(\Box\neg A \land A)$). Hence it turns out that the modality $\Box$ has the properties of modal system G, in which the accessibility relation is transitive and does not have infinite ascending chains.

Since we have introduced boxed formulas for capturing a notion of minimality among worlds, in the rest of this work we will only use this modality in front of negated formulas. Hence, to be precise, the language $\mathcal{L}_R$ of our tableau extends $\mathcal{L}$ with modal formulas of the form $\Box\neg A$.

3.2.2 Preferential Logic P

The axiomatization of **P** can be obtained from the axiomatization of **R** by removing the axiom RM. The resulting axiom system is shown in Figure 3.2.
As for **R**, the semantics of **P** is defined by considering possible world structures with a preference relation (an irreflexive and transitive relation), which is no longer assumed to be modular.

REF. $A \mid\!\sim A$

LLE. If $\vdash_{PC} A \leftrightarrow B$, then $\vdash (A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$

RW. If $\vdash_{PC} A \rightarrow B$, then $\vdash (C \mid\!\sim A) \rightarrow (C \mid\!\sim B)$

CM. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \wedge B \mid\!\sim C)$

AND. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \mid\!\sim B \wedge C)$

OR. $((A \mid\!\sim C) \wedge (B \mid\!\sim C)) \rightarrow (A \vee B \mid\!\sim C)$

Figure 3.2: Axioms and rules of KLM logic **P**. We use $\vdash_{PC}$ to denote provability in the propositional calculus, whereas $\vdash$ is used to denote provability in **P**.

Definition 3.9 (Semantics of P, Definition 16 in [KLM90])**.** *A preferential model is a triple*

$$\mathcal{M} = \langle \mathcal{W}, <, V \rangle$$

where $\mathcal{W}$ and V are defined as for rational models in Definition 3.1, and $<$ is an irreflexive and transitive relation on $\mathcal{W}$. The truth conditions for a formula F, the smoothness condition, and the notions of validity of a formula are defined as for rational models in Definition 3.1.

As for rational models, we have extended the definition of preferential models given by KLM in order to deal with boolean combinations of formulas.

We define the satisfiability of conditional formulas with respect to worlds rather than with respect to models for uniformity sake. As for **R**, by the transitivity of $<$, the smoothness condition is equivalent to the strong smoothness condition, corresponding to the finite chain condition for $<$.

Here again, we consider the language $\mathcal{L}_P$ of the calculus introduced in Section 5.2; $\mathcal{L}_P$ corresponds to the language $\mathcal{L}_R$, i.e. it extends $\mathcal{L}$ by boxed formulas of the form $\Box\neg A$. It follows that, even in **P**, we can prove that, for any formula A, $w \in Min_<(A)$ iff $\mathcal{M}, w \models A \wedge \Box\neg A$.

Multi-linear models for P

In Chapter 5 we will need a special kind of preferential models, that we call *multi-linear*. As we will see, these models will be useful in order to provide an optimal calculus for **P**. Indeed, as we will see in Section 5.2.1, our calculus for **P** based on multi-linear models will allow us to define proof search procedures for testing the satisfiability of a set of formulas in **P** in nondeterministic polynomial time. This result matches the known complexity results for **P**, according to which the problem of validity for **P** is in coNP.

Definition 3.10. *A finite preferential model $\mathcal{M} = (\mathcal{W}, <, V)$ is multi-linear if the set of worlds $\mathcal{W}$ can be partitioned into a set of components $\mathcal{W}_i$ for $i = 1, \ldots, n$, that is $\mathcal{W} = \mathcal{W}_1 \cup \ldots \cup \mathcal{W}_n$ and*

1. *the relation $<$ is a total order on each $\mathcal{W}_i$;*

2. *the elements in two different components $\mathcal{W}_i$ and $\mathcal{W}_j$ are incomparable with respect to $<$.*

The following theorem shows that we could restrict our consideration to multi-linear models and generalizes Lemma 8 in [LM92].

Theorem 3.11. *Let Γ be a set of formulas, if Γ is satisfiable with respect to the logic* **P***, then it has a multi-linear model.*

Proof. Let us make explicit the negated conditionals in Γ by rewriting it as

$$\Gamma = \Gamma', \neg(C_1 \mid\!\sim D_1), \ldots, \neg(C_k \mid\!\sim D_k).$$

Assume Γ is satisfiable, then there is a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ and $x \in \mathcal{W}$, such that $\mathcal{M}, x \models \Gamma$. We have that there are $y_1, \ldots, y_k \in \mathcal{W}$, such that for each $j = 1, \ldots, k$,

$$y_j \in Min_<(C_j) \text{ and } \mathcal{M}, y_j \not\models D_j$$

We define for x and each y_j:

$$\mathcal{W}_x = \{z \in \mathcal{W} \mid z < x\} \cup \{x\}$$

$$\mathcal{W}_{y_j} = \{z \in \mathcal{W} \mid z < y_j\} \cup \{y_j\}$$

Moreover, we consider for each $j = 1, \ldots, k$ a renaming function (i.e. a bijection) f_j whose domain is $\mathcal{W}_{y_j}$ that makes a copy $\mathcal{W}_{f_j(y_j)}$ of $\mathcal{W}_{y_j}$ which is (i) disjoint from $\mathcal{W}_x$, (ii) disjoint from any $\mathcal{W}_{y_l}$, and (iii) disjoint from any other $\mathcal{W}_{f_l(y_l)}$ with $l \neq j$. Observe that we make k disjoint sets $\mathcal{W}_{f_j(y_j)}$ even if some y_j's coincide among themselves or coincide with x. We define a model $\mathcal{M}' = \langle \mathcal{W}', <', V' \rangle$ as follows:

$$\mathcal{W}' = \mathcal{W}_x \cup \mathcal{W}_{f_1(y_1)} \ldots \mathcal{W}_{f_k(y_k)}$$

The relation $<'$ is defined as follows:

$u <' v$ iff

(i) $u, v \in \mathcal{W}_x$ and $u < v$
or
(ii) $u, v \in \mathcal{W}_{f_j(y_j)}$ so that $u = f_j(z)$ and $v = f_j(w)$, where $z, w \in \mathcal{W}_{y_j}$ and $z < w$.

Observe that elements in different components (i.e. $\mathcal{W}_x$ or $\mathcal{W}_{y_j}$) are incomparable with respect to $<'$.

Finally, we let $V'(z) = V(z)$ for $z \in \mathcal{W}_x$ and for $u \in \mathcal{W}_{f_j(y_j)}$ with $u = f_j(w)$, we let $V'(u) = V(w)$.

We prove that $\mathcal{M}', x \models \Gamma$. The claim is obvious for propositional formulas and for (negated) boxed formulas by definition of $\mathcal{W}_x$.

For any negated conditional $\neg(C_j \mid\!\sim D_j)$, we have that $y_j \in Min_<(C_j)$ and $\mathcal{M}, y_j \not\models D_j$. By definition of $\mathcal{M}'$ we get that $\mathcal{M}', f_j(y_j) \models C_j$ and $\mathcal{M}', f_j(y_j) \not\models D_j$; we have to show that there is no $u \in \mathcal{W}_{f_j(y_j)}$ such that $u <' f_j(y_j)$ and $\mathcal{M}', u \models C_j$. But if there were a such u, we would get that $u = f_j(z)$ for some $z \in \mathcal{W}_{y_j}$ with $z < y_j$ and $\mathcal{M}, z \models C_j$ against the minimality of y_j.

For any positive conditional in Γ, say $E \mid\!\sim F$, let $u \in Min_<(E)$: if $u \in \mathcal{W}_x$ it must be $u \in Min_<(E)$ thus $\mathcal{M}', u \models F$. If $u \in \mathcal{W}_{f_j(y_j)}$, then $u = f_j(z)$ for some

$z \in \mathcal{W}_{y_j}$; it must be $z \in Min_<(E)$, for otherwise if it were $z' < z$ with $\mathcal{M}, z' \models E$, since $z' \in \mathcal{W}_{y_j}$ we would have $f_j(z') < u$ and $\mathcal{M}, f_j(z') \models E$ against the minimality of u. Thus $z \in Min_<(E)$ and then $\mathcal{M}, z \models F$, and this implies $\mathcal{M}', u \models F$.

We now define a multi-linear model $\mathcal{M}_1 = \langle \mathcal{W}', <_1, V' \rangle$ as follows: we let $<_1$ be any total order on $\mathcal{W}_x$ and on each $\mathcal{W}_{f_j(y_j)}$ which respects $<'$; the elements in different components remain incomparable. More precisely $<_1$ satisfies:

- if $u <' v$ then $u <_1 v$
- for each $u, v \in \mathcal{W}_x$ $(u, v \in \mathcal{W}_{f_j(y_j)})$ with $u \neq v$, $u <_1 v$ or $v <_1 u$
- for each $u \in \mathcal{W}_x$, $v \in \mathcal{W}_{f_j(y_j)}, u \not<_1 v$ and $v \not<_1 u$
- for each $u \in \mathcal{W}_{f_i(y_i)}, v \in \mathcal{W}_{f_j(y_j)}$, with $i \neq j$ $u \not<_1 v$ and $v \not<_1 u$

In Figure 3.4 we show an example of multi-linear model, obtained by applying the above construction to the model represented in Figure 3.3.

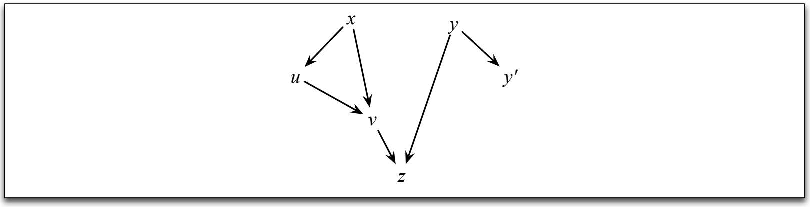


Figure 3.3: A preferential model satisfying a set of formulas Γ. Edges represent the preference relation $<$ $(u < x, v < u$, and so on).

We show that $\mathcal{M}_1, x \models \Gamma$. For propositional formulas the claim is obvious. For positive boxed-formulas we have: if $\Box\neg A \in \Gamma$, and $z <_1 x$, then $z \in \mathcal{W}_x$, thus $z <' x$, and the result follows by $\mathcal{M}', x \models \Gamma$. For negated boxed formulas, we similarly have: $\neg\Box\neg A \in \Gamma$, then $\mathcal{M}', x \models \neg\Box\neg A$, thus there exists $z <' x$ such that $\mathcal{M}', z \models A$, but $z <' x$ implies $z <_1 x$ and we can conclude.

For negated conditionals, let $\neg(C_j \mid\!\sim D_j)$, we know that $\mathcal{M}', x \models \neg(C_j \mid\!\sim D_j)$, witnessed by the C_j-minimal element $f_j(y_j)$. Since the propositional evaluation is the same, we only have to check that $f_j(y_j)$ is also minimal with respect to $<_1$. Suppose

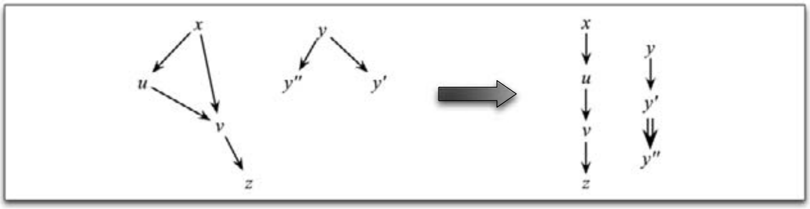


Figure 3.4: A multi-linear model obtained by means of the construction described in the proof of Theorem 3.11. Edges represent the preference relation $<_1$. An additional edge $\Downarrow$ has been added to let $<_1$ be a total order. In order to have that elements in different components are incomparable with respect to $<_1$, in the rightmost component the world z has been renamed in y''.

REF. $A \mid\!\sim A$

LLE. If $\vdash_{PC} A \leftrightarrow B$, then $\vdash (A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$

RW. If $\vdash_{PC} A \rightarrow B$, then $\vdash (C \mid\!\sim A) \rightarrow (C \mid\!\sim B)$

CM. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \wedge B \mid\!\sim C)$

AND. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \mid\!\sim B \wedge C)$

LOOP. $(A_0 \mid\!\sim A_1) \wedge (A_1 \mid\!\sim A_2)...(A_{n-1} \mid\!\sim A_n) \wedge (A_n \mid\!\sim A_0) \rightarrow (A_0 \mid\!\sim A_n)$

CUT. $((A \mid\!\sim B) \wedge (A \wedge B \mid\!\sim C)) \rightarrow (A \mid\!\sim C)$

Figure 3.5: Axioms and rules of KLM logic **CL**. We use $\vdash_{PC}$ to denote provability in the propositional calculus, whereas $\vdash$ is used to denote provability in **CL**.

it is not, then there is $z \in \mathcal{W}_{f_j(y_j)}$ with $z <_1 f_j(y_j)$ such that $\mathcal{M}_1, z \models C_j$, but we would get $z <' f_j(y_j)$ against the minimality of $f_j(y_j)$.

For positive conditionals in Γ, say $E \mid\!\sim F$, let $u \in Min_{<_1}(E)$. It must be also $u \in Min_{<_1}(E)$, for otherwise, if there were $v <' u$, such that $\mathcal{M}', v \models E$ then we would get also $v <_1 u$ and $\mathcal{M}_1, v \models E$, against the minimality of u in $\mathcal{M}_1$.

■

3.2.3 Loop Cumulative Logic CL

The next KLM logic we consider is **CL**, weaker than **P**. The axiomatization of **CL** can be obtained from the axiomatization of **P** by removing the axiom OR and by adding the following infinite set of LOOP axioms:

LOOP. $(A_0 \mid\!\sim A_1) \wedge (A_1 \mid\!\sim A_2)...(A_{n-1} \mid\!\sim A_n) \wedge (A_n \mid\!\sim A_0) \rightarrow (A_0 \mid\!\sim A_n)$

and the following axiom CUT:

CUT. $((A \mid\!\sim B) \wedge (A \wedge B \mid\!\sim C)) \rightarrow (A \mid\!\sim C)$

The resulting axiom system is presented in Figure 3.5. Notice that the axioms (CUT) and (LOOP) are derivable in **P** (and therefore in **R**).
The following Definition is essentially the same as Definition 13 in [KLM90], but it is extended to boolean combinations of conditionals.

Definition 3.12 (Semantics of **CL**). *A loop-cumulative model is a tuple*

$$\mathcal{M} = \langle S, \mathcal{W}, l, <, V \rangle$$

where:

- *S is a set, whose elements are called states;*
- *$\mathcal{W}$ is a set of possible worlds;*

- *$l : S \mapsto 2^{\mathcal{W}}$ is a function that labels every state with a nonempty set of worlds;*
- *$<$ is an irreflexive and transitive relation on S;*
- *V is a valuation function $V : \mathcal{W} \longmapsto 2^{ATM}$, which assigns to every world w the atoms holding in that world.*

For $s \in S$ and A propositional, we let $\mathcal{M}, s \models\!\equiv A$ if $\forall w \in l(s)$, $\mathcal{M}, w \models A$, where $\mathcal{M}, w \models A$ is defined as in propositional logic. Let $Min_<(A)$ be the set of minimal states s such that $\mathcal{M}, s \models\!\equiv A$. We define $\mathcal{M}, s \models\!\equiv A \mid\!\sim B$ if $\forall s' \in Min_<(A)$, $\mathcal{M}, s' \mid \equiv B$. The relation $\models\!\equiv$ can be extended to boolean combinations of conditionals in the standard way. We assume that $<$ satisfies the smoothness condition.

The above notion of cumulative model extends the one given by KLM to boolean combinations of conditionals. A further extension to arbitrary boolean combinations will be provided by the notion of CL-preferential model below.

Here again, we define satisfiability of conditionals with respect to states rather than with respect to models for uniformity reasons. Indeed, a conditional is satisfied by a state of a model only if and only if it is satisfied by all the states of that model, hence by the whole model.

As for **P** and **R**, by the transitivity of $<$, the smoothness condition is equivalent to the Strong smoothness condition. In turn, this entails that $<$ does not have infinite descending chains.

We show that we can map loop-cumulative models into preferential models extended with an additional accessibility relation R. We call these preferential models *CL-preferential models*. The idea is to represent states as sets of possible worlds related by R in such a way that a formula is satisfied in a state s just in case it is satisfied in all possible worlds w' accessible from a world w corresponding to s. The syntactic counterpart of the extra accessibility relation R is a modality L. Given a loop-cumulative model $\mathcal{M}$ and the corresponding CL-preferential model $\mathcal{M}'$, $\mathcal{M}, s \models\!\equiv A$ iff for a world $w \in \mathcal{M}'$ corresponding to s, we have that $\mathcal{M}', w \models LA$. As we will see, this mapping enables us to use a variant of the tableau calculus for **P** to deal with system **CL**. As for **P**, the tableau calculus for **CL** will use boxed formulas. In addition, it will also use L-formulas. Thus, the formulas that appear in the tableaux for **CL** belong to the language $\mathcal{L}_L$ obtained from $\mathcal{L}$ as follows: (i) if A is propositional, then $A \in \mathcal{L}_L$; $LA \in \mathcal{L}_L$; $\Box\neg LA \in \mathcal{L}_L$; (ii) if A, B are propositional, then $A \mid\!\sim B \in \mathcal{L}_L$; (iii) if F is a boolean combination of formulas of $\mathcal{L}_L$, then $F \in \mathcal{L}_L$. Observe that the only allowed combination of $\Box$ and L is in formulas of the form $\Box\neg LA$ where A is propositional.

We can map loop-cumulative models into preferential models with an additional accessibility relation as defined below:

Definition 3.13 (CL-preferential models)**.** *A CL-preferential model has the form*

$$\mathcal{M} = \langle \mathcal{W}, R, <, V \rangle$$

where:

- *$\mathcal{W}$ and V are defined as for preferential models in Definition 3.9;*
- *$<$ is an irreflexive and transitive relation on $\mathcal{W}$;*
- *R is a serial relation on $\mathcal{W}$;*

We add to the truth conditions for preferential models in Definition 3.9 the following clause:

$$\mathcal{M}, w \models LA \text{ if, for all } w', \ wRw' \text{ implies } \mathcal{M}, w' \models A$$

The relation $<$ satisfies the following smoothness condition*: if $\mathcal{M}, w \models LA$, then $w \in Min_<(LA)$ or $\exists w' \in Min_<(LA)$ such that $w' < w$.*
Moreover, we write $\mathcal{M}, w \models A \mathrel{|\!\sim} B$ if for all $w' \in Min_<(LA)$ we have $\mathcal{M}, w' \models LB$.

We can prove that, given a loop-cumulative model $\mathcal{M} = \langle S, \mathcal{W}, l, <, V\rangle$ satisfying a boolean combination of conditional formulas, one can build a CL-preferential model $\mathcal{M}' = \langle \mathcal{W}', R, <', V'\rangle$ satisfying the same combination of conditionals. *Vice versa*, given a CL-preferential model $\mathcal{M} = \langle \mathcal{W}, R, <, V\rangle$ satisfying a boolean combination of conditional formulas, one can build a loop-cumulative model $\mathcal{M}' = \langle S, \mathcal{W}, l, <', V'\rangle$ satisfying the same combination of conditional formulas. This is stated in a rigorous manner by the following proposition:

Proposition 3.14. *A boolean combination of conditional formulas is satisfiable in a loop-cumulative model $\mathcal{M} = \langle S, \mathcal{W}, l, <, V\rangle$ iff it is satisfiable in a CL-preferential model $\mathcal{M}' = \langle \mathcal{W}', R, <', V'\rangle$.*

Proof. The Proposition immediately follows from the following Lemma:

Lemma 3.15. *A set of conditional formulas $\{(\neg)A_1 \mathrel{|\!\sim} B_1, \ldots, (\neg)A_n \mathrel{|\!\sim} B_n\}$ is satisfiable in a loop-cumulative model $\mathcal{M} = \langle S, \mathcal{W}, l, <, V\rangle$ iff it is satisfiable in a CL-preferential model $\mathcal{M}' = \langle \mathcal{W}', R, <', V'\rangle$.*

First, we prove the *only if* direction. Let $\mathcal{M} = \langle S, \mathcal{W}, l, <, V\rangle$ be a loop-cumulative model, and $s \in S$ s.t. $(\mathcal{M}, s) \models\!\equiv \{(\neg)A_i \mathrel{|\!\sim} B_i\}$.

We build a CL-preferential model $\mathcal{M}' = \langle \mathcal{W}', R, <', V'\rangle$ as follows:

- $\mathcal{W}' = \{(s, w) : s \in S \text{ and } w \in l(s)\}$;
- $(s, w)R(s, w')$ for all $(s, w), (s, w') \in \mathcal{W}'$;
- $(s, w) <' (s', w')$ if $s < s'$;
- $V'(s, w) = V(w)$.

Observe that for each $s \in S$ there is at least one corresponding $(s, w) \in \mathcal{W}'$, since $l(s) \neq \emptyset$. From the fact that $<$ in $\mathcal{M}$ is irreflexive and transitive it immediately follows by construction that also $<'$ in $\mathcal{M}'$ satisfies the same properties. We show in Fact 3.20 below that $<'$ satisfies the smoothness condition on L-formulas. The relation R is serial, since it is reflexive.

Fact 3.16. *For every propositional formula A we have that $(\mathcal{M}, s) \models\!\equiv A$ if and only if $(\mathcal{M}', (s, w)) \models LA$.*

Proof of Fact 3.16. ($\Rightarrow$) Let $(\mathcal{M}, s) \models\!\equiv A$. By definition, for all $w \in l(s)$, $(\mathcal{M}, w) \models A$. By induction on the complexity of A, we can easily show that $(\mathcal{M}', (s, w)) \models A$. Since $R(s, w) = \{(s, w') \mid w' \in l(s)\}$, it follows that $(\mathcal{M}', (s, w)) \models LA$.
($\Leftarrow$) Let $(\mathcal{M}', (s, w)) \models LA$. Then, for all $(s, w') \in R(s, w)$, $(\mathcal{M}', (s, w')) \models A$. By definition of $\mathcal{M}'$ it follows that for all $w' \in l(s)$, $(\mathcal{M}, w') \models A$. Hence, $(\mathcal{M}, s) \models\!\equiv A$.

□ *Fact 3.16*

Fact 3.17. *$s \in Min_<(A)$ in $\mathcal{M}$ iff $(s, w) \in Min_{<'}(LA)$ in $\mathcal{M}'$.*

Proof of Fact 3.17. ($\Rightarrow$) Let $s \in Min_<(A)$ in $\mathcal{M}$. Consider (s, w) in $\mathcal{M}'$. By Fact 3.16, $(\mathcal{M}', (s, w)) \models LA$. By absurd, suppose there exists a (s', w') s.t. $(\mathcal{M}', (s', w')) \models LA$, and $(s', w') < (s, w)$. By Fact 3.16, $(\mathcal{M}, s') \mid\!\equiv A$, and $s' < s$ by construction, which contradicts the fact that $s \in Min_<(A)$ in $\mathcal{M}$. Hence $(s, w) \in Min_{<'}(LA)$ in $\mathcal{M}'$.
($\Leftarrow$) Let $(s, w) \in Min_{<'}(LA)$ in $\mathcal{M}'$. Consider s in $\mathcal{M}$. By Fact 3.16, $(\mathcal{M}, s) \mid\!\equiv A$. Furthermore there is no $s' < s$ s.t. $(\mathcal{M}, s') \mid\!\equiv A$. By absurd suppose there was such a s'. By construction of $\mathcal{M}'$, and by Fact 3.16, $(\mathcal{M}', (s', w')) \models LA$, and $(s', w') < (s, w)$, which is a contradiction. Hence $s \in Min_<(A)$.

□ *Fact 3.17*

Fact 3.18. *For every conditional formula $A \mid\!\sim B$ we have that $(\mathcal{M}, s) \mid\!\equiv A \mid\!\sim B$ iff $(\mathcal{M}', (s, w)) \models A \mid\!\sim B$.*

Proof of Fact 3.18. ($\Rightarrow$) Let $(\mathcal{M}, s) \mid\!\equiv A \mid\!\sim B$. Then for all $s' \in Min_<(A)$, $(\mathcal{M}, s') \mid\!\equiv B$. By Facts 3.16 and 3.17, it follows that for all $(s', w') \in Min_{<'}(LA)$, $(\mathcal{M}', (s', w')) \models LB$, hence $(\mathcal{M}', (s, w)) \models A \mid\!\sim B$.
($\Leftarrow$) Let $(\mathcal{M}', (s, w)) \models A \mid\!\sim B$. Then, for all $(s', w') \in Min'_<(LA)$, $(\mathcal{M}', (s', w')) \models LB$. By Facts 3.16 and 3.17 it follows that for all $s' \in Min_<(A)$ in $\mathcal{M}$, $(\mathcal{M}, s') \mid\!\equiv B$. Hence, $(\mathcal{M}, s) \mid\!\equiv A \mid\!\sim B$.

□ *Fact 3.18*

Fact 3.19. *For every negated conditional formula $\neg(A \mid\!\sim B)$ we have that $(\mathcal{M}, s) \mid\!\equiv \neg(A \mid\!\sim B)$ iff $(\mathcal{M}', (s, w)) \models \neg(A \mid\!\sim B)$.*

Proof of Fact 3.19. ($\Rightarrow$) Let $(\mathcal{M}, s) \mid\!\equiv \neg(A \mid\!\sim B)$. Then there is an $s' \in Min_<(A)$ s.t. $\mathcal{M}, s' \not\mid\!\equiv B$. Consider (s', w') in $\mathcal{M}'$. By Facts 3.16 and 3.17 $(s', w') \in Min_{<'}(LA)$ and $(\mathcal{M}', (s', w')) \not\models LB$. Hence, $(\mathcal{M}', (s, w)) \models \neg(A \mid\!\sim B)$.
($\Leftarrow$) Let $(\mathcal{M}', (s, w)) \models \neg(A \mid\!\sim B)$. Then, there is a $(s', w') \in Min'_<(LA)$ s.t. $(\mathcal{M}', (s', w')) \not\models LB$. Consider s' in $\mathcal{M}$. From Facts 3.16 and 3.17 we conclude that $s' \in Min_<(A)$, and $(\mathcal{M}, s') \not\mid\!\equiv B$. Hence $(\mathcal{M}, s) \mid\!\equiv \neg(A \mid\!\sim B)$.

□ *Fact 3.19*

From Facts 3.18 and 3.19 we conclude that $(\mathcal{M}', (s, w)) \models \{(\neg)A_i \mid\!\sim B_i\}$. Furthermore, we show that $<'$ satisfies the smoothness condition on L-formulas.

Fact 3.20. *$<'$ satisfies the smoothness condition on L-formulas.*

Proof of Fact 3.20. Let $(\mathcal{M}', (s, w)) \models LA$, and $(s, w) \notin Min_{<'}(LA)$. By Fact 3.16 $(\mathcal{M}, s) \mid\!\equiv A$, and by Fact 3.17, $s \notin Min_<(A)$ in $\mathcal{M}$. By the smoothness condition in $\mathcal{M}$ there is s' such that $s' \in Min_<(A)$ and $s' < s$. Consider any $(s', w') \in \mathcal{M}'$. By Fact 3.17 $(s', w') \in Min_{<'}(LA)$, and by definition of $<'$, $(s', w') <' (s, w)$.

□ *Fact 3.20*

Let us now consider the *if* direction. Let the set of conditionals $\{(\neg)A_i \mid\!\sim B_i\}$ be satisfied in a possible world w in the CL-preferential model $\mathcal{M} = \langle \mathcal{W}, R, <, V \rangle$. We build a Loop-Cumulative model $\mathcal{M}' = \langle S, \mathcal{W}, l, <', V' \rangle$ as follows (Rw is defined as $Rw = \{w' \in \mathcal{W} \mid (w, w') \in R\}$):

- $S = \{(w, Rw) \mid w \in \mathcal{W}\}$;
- $l((w, Rw)) = Rw$;
- $(w, Rw) <' (w', Rw')$ if $w < w'$;
- $V'(w) = V(w)$.

From the fact that $<$ in $\mathcal{M}$ is transitive and irreflexive, it follows by construction that $<'$ in $\mathcal{M}'$ is transitive and irreflexive. As far as the smoothness condition, see Fact 3.24 below. Furthermore, for all $(w, Rw) \in S$, $l(w, Rw) \neq \emptyset$, since R is serial.

We now show that $(\mathcal{M}', (w, Rw)) \models\equiv \{(\neg)A_i \mid\sim B_i\}$.

Fact 3.21. *For every propositional formula A we have that $(\mathcal{M}, w) \models LA$ if and only if $(\mathcal{M}', (w, Rw)) \models\equiv A$.*

Proof of Fact 3.21. ($\Rightarrow$) Let $(\mathcal{M}, w) \models LA$. Then for all $w' \in Rw$, $\mathcal{M}, w' \models A$. By definition of $\mathcal{M}'$, it follows that for all $w' \in l(w, Rw)$, $\mathcal{M}', w' \models A$. Hence $(\mathcal{M}', (w, Rw)) \models\equiv A$.
($\Leftarrow$) Let $(\mathcal{M}', (w, Rw)) \models\equiv A$. Then, for all $w' \in l(w, Rw)$, $(\mathcal{M}', w') \models A$. By induction on A, we show that for all $w' \in Rw$, $(\mathcal{M}, w') \models A$, hence $(\mathcal{M}, w) \models LA$.

□ *Fact 3.21*

Fact 3.22. *$w \in Min_<(LA)$ in $\mathcal{M}$ iff $(w, Rw) \in Min_{<'}(A)$ in $\mathcal{M}'$.*

Proof of Fact 3.22. ($\Rightarrow$) Let $w \in Min_<(LA)$ in $\mathcal{M}$. Consider (w, Rw). By Fact 3.21 $(\mathcal{M}', (w, Rw)) \models\equiv A$. Furthermore, suppose by absurd there was (w', Rw') s.t. $(\mathcal{M}', (w', Rw')) \models\equiv A$, and $(w', Rw') <' (w, Rw)$. Then in $\mathcal{M}$, $\mathcal{M}, w' \models LA$ and $w' < w$, which contradicts the fact that $w \in Min_<(LA)$. It follows that in $\mathcal{M}'$, $(w, Rw) \in Min_{<'}(A)$.
($\Leftarrow$) Let $(w, Rw) \in Min_{<'}(A)$ in $\mathcal{M}'$. Consider w in $\mathcal{M}$. By Fact 3.21, $(\mathcal{M}, w) \models LA$. By absurd, suppose there was a w' s.t. $\mathcal{M}, w' \models LA$ and $w' < w$. By Fact 3.21, $(\mathcal{M}', (w', Rw')) \models\equiv A$, and $(w', Rw') <' (w, Rw)$, which contradicts the fact that $(w, Rw) \in Min_{<'}(A)$. It follows that $w \in Min_<(LA)$ in $\mathcal{M}$.

□ *Fact 3.22*

We can reason similarly to what done in Facts 3.18 and 3.19 above to prove the following Fact:

Fact 3.23. *For every conditional formula $(\neg)A \mid\sim B$ we have that $(\mathcal{M}, w) \models (\neg)A \mid\sim B$ iff $(\mathcal{M}', (w, Rw)) \models\equiv (\neg)A \mid\sim B$.*

We conclude that $(\mathcal{M}', (w, Rw)) \models\equiv \{(\neg)A_i \mid\sim B_i\}$. Furthermore, we show that $<'$ satisfies the smoothness condition:

Fact 3.24. *$<'$ satisfies the smoothness condition on L-formulas.*

Proof of Fact 3.24. Let $(\mathcal{M}', (w, Rw)) \models\equiv A$ and $(w, Rw) \not\in Min_{<'}(A)$ in $\mathcal{M}'$. By Facts 3.21 and 3.22, $(\mathcal{M}, w) \models LA$ and $w \not\in Min_<(LA)$ in $\mathcal{M}$. By the smoothness condition on L-formulas in $\mathcal{M}$, it follows that in $\mathcal{M}$ there is $w' < w$ s.t. $(\mathcal{M}, w') \models LA$ and $w' \in Min_<(LA)$. Consider (w', Rw') in $\mathcal{M}'$. By Facts 3.21 and 3.22 $(\mathcal{M}', (w', Rw')) \mid \equiv A$ and $(w', Rw') \in Min_{<'}(A)$ in $\mathcal{M}'$. □ *Fact 3.24*

■

REF. $A \mid\!\sim A$

LLE. If $\vdash_{PC} A \leftrightarrow B$, then $\vdash (A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$

RW. If $\vdash_{PC} A \rightarrow B$, then $\vdash (C \mid\!\sim A) \rightarrow (C \mid\!\sim B)$

CM. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \wedge B \mid\!\sim C)$

AND. $((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \mid\!\sim B \wedge C)$

CUT. $((A \mid\!\sim B) \wedge (A \wedge B \mid\!\sim C)) \rightarrow (A \mid\!\sim C)$

Figure 3.6: Axioms and rules of KLM logic **C**. We use $\vdash_{PC}$ to denote provability in the propositional calculus, whereas $\vdash$ is used to denote provability in **C**.

Similarly to what done for **P**, we can define multi-linear CL-preferential models as follows:

Definition 3.25. *A finite CL-preferential model $\mathcal{M} = (\mathcal{W}, R, <, V)$ is multi-linear if the set of worlds $\mathcal{W}$ can be partitioned into a set of components $\mathcal{W}_i$ for $i = 1, \ldots, n$ (that is $\mathcal{W} = \mathcal{W}_1 \cup \ldots \cup \mathcal{W}_n$), and in each $\mathcal{W}_i$:*

1. *there is a totally ordered chain of worlds $w_1, w_2, \ldots, w_h$ with respect to $<$ (i.e. $w_1 < w_2 < \cdots < w_h$) such that all other worlds $w \in \mathcal{W}_i$ are R-accessible from some w_l in the chain, i.e. $\forall w \in \mathcal{W}_i$ such that $w \neq w_1, w_2, \ldots, w_h$, there is w_i, $i = 1, 2, \ldots, h$ such that $w_i R w$.*
2. *for all w', w'', w''', if $w' < w''$ and $w'' R w'''$, then $w' < w'''$.*

Moreover, the elements of different $\mathcal{W}_i$ are incomparable with respect to $<$.

Observe that the worlds in each $\mathcal{W}_i$ are partitioned into two sets of worlds $\mathcal{W}_i^1$ and $\mathcal{W}_i^2$. Worlds in $\mathcal{W}_i^1$ are organized in a totally ordered chain, whereas worlds in $\mathcal{W}_i^2$ are reachable from those in $\mathcal{W}_i^1$ through R. Similarly to the case of **P**, we can prove the following theorem:

Theorem 3.26. *Let Γ be any set of formulas, if Γ is satisfiable in a CL-preferential model, then it has a multi-linear CL-preferential model.*

3.2.4 Cumulative Logic C

The weakest logical system considered by KLM [KLM90] is Cumulative Logic **C**. System **C** is weaker than **CL** considered above since it does not have the set of (LOOP) axioms. The axiom system of **C**is presented in Figure 3.6.
At a semantic level, the difference between **CL** models and **C** models is that in **CL** models the relation $<$ is transitive, whereas in **C** it is not. Thus, cumulative **C** models are defined as follows:

Definition 3.27 (Semantics of **C**, Definitions 5, 6, 7 in [KLM90])**.** *A cumulative model is a tuple*

$$\mathcal{M} = \langle S, \mathcal{W}, l, <, V \rangle$$

where S, $\mathcal{W}$, l, and V are defined as for loop-cumulative models in Definition 3.12, whereas $<$ is an irreflexive relation on S. The truth definitions of formulas are as for loop-cumulative models in Definition 3.12. We assume that $<$ satisfies the smoothness condition.

Since $<$ is not transitive, we can no longer show that the smoothness condition is equivalent to the strong smoothness condition; furthermore, $<$ may have infinite descending chains. As a matter of fact, in **C** there can be sets of formulas which can be satisfied only by models containing infinite descending chains (or cycles, if the model is finite). As an example, consider the following set of formulas: $\Gamma = \{\neg(C \mid\sim B), C \mid\sim A, A \mid\sim B, B \mid\sim C\}$. This is an instance of the negation of (LOOP). Γ can be satisfied by a **C** model just in case it contains a cycle $w_1, w_2, w_3, w_1, w_2, w_3, \ldots$, such that w_1 is a minimal C-world, w_2 a minimal A-world, w_3 a minimal B-world; in turn w_3 must be preceded again by a minimal C-world as w_1, and so on. In case of a finite **C** model, Γ can be satisfied only if the model contains cycles, made out of the same sequence of worlds. On the contrary, it is easy to see that this model does not satisfy the strong smoothness condition.

Similarly to what we have done for loop-cumulative models, we can establish a correspondence between cumulative models and preferential models augmented with an accessibility relation in which the preference relation $<$ is an irreflexive relation satisfying the smoothness condition. We call these models C-preferential models.

Definition 3.28 (C-preferential models). *A C-preferential model has the form $\mathcal{M} = \langle \mathcal{W}, R, <, V \rangle$ where: $\mathcal{W}$ is a non-empty set of items called worlds; R is a serial accessibility relation; $<$ is an irreflexive relation on $\mathcal{W}$ satisfying the smoothness condition for L-formulas; V is a function $V : \mathcal{W} \longmapsto 2^{ATM}$, which assigns to every world w the atomic formulas holding in that world. The truth conditions for the boolean cases are defined in the obvious way. Truth conditions for modal and conditional formulas are the same as in CL-preferential models in Definition 3.13, thus:*

- $M, w \models LA$ *if for all* w', wRw' *implies* $\mathcal{M}, w' \models A$
- $\mathcal{M}, w \models A \mid\sim B$ *if for all* $w' \in Min_<(LA)$, *we have* $\mathcal{M}, w' \models LB$.

The correspondence between cumulative and preferential models is established by the following proposition. Its proof is the same as the proof of Proposition 3.14 (except for transitivity) and is therefore omitted.

Proposition 3.29. *A boolean combination of conditional formulas is satisfiable in a cumulative model $\mathcal{M} = \langle S, \mathcal{W}, l, <, V \rangle$ iff it is satisfiable in a C-preferential model $\mathcal{M}' = \langle \mathcal{W}', R, <', V' \rangle$.*

3.3 Conditional Logics vs KLM Logics

We conclude this chapter by discussing the relationship between preferential logics of the KLM framework and conditional logics introduced in Chapter 2. As mentioned

above, KLM logics are equivalent to the flat fragment, i.e. without nested conditionals, of some well known conditional logics.
The close connection between conditional logics and KLM logics is immediately suggested by comparing some conditional axioms/rules with some axioms/rules characterizing the KLM logics. It is easy to observe how the following axioms are similar at glance:

Conditional logic	**KLM logic**
ID $A \Rightarrow A$	REF $A \mid\!\sim A$
RCEA $\begin{array}{c} \vdash A \leftrightarrow B \\ \vdash (A \Rightarrow C) \rightarrow (B \Rightarrow C) \end{array}$	LLE If $\vdash_{PC} A \leftrightarrow B$, then $\vdash (A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$
CV $((A \Rightarrow B) \wedge \neg(A \Rightarrow \neg C)) \rightarrow ((A \wedge C) \Rightarrow B)$	RM $((A \mid\!\sim B) \wedge \neg(A \mid\!\sim \neg C)) \rightarrow ((A \wedge C) \mid\!\sim B)$

The connection between conditional logics and nonmonotonic inference relations has been first noted by Van Benthem in [Sho87a], then it has been studied by Bell [Bel91] and by KLM in [KLM90]. Roughly speaking, it has been observed that the conditional operator $\Rightarrow$ encodes a nonmonotonic deduction relation. However, in [KLM90] and [Bel91], the authors remark the following two main difficulties in considering classes of nonmonotonic inference systems as only a typographical variant of conditional logics, in which $\mid\!\sim$ is replaced by $\Rightarrow$:

1. conditionals can be nested, whereas nested $\mid\!\sim$ do not have any meaning;
2. the conditional semantics evaluation refers to the actual world whereas this does not make sense in the KLM framework.

A fundamental contribution on the study of the connection between conditional logics and preferential logics has been given by Crocco and Lamarre. In [CL92], they remark that the above two problems are strictly related, by the fact that if the language of conditional logics is restricted to its flat fragment, keeping or dropping the actual world does not change anything. Then, they give a complete and general proof of the equivalence between KLM and conditional logics, such that, given a definition of a nonmonotonic class of inference relations, it is possible to obtain the exact corresponding conditional logic. Let us analyze their result in more detail.

Crocco and Lamarre denote a property p of a nonmonotonic inference relation as a rule having the following form:

$$\begin{array}{c} \alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_n \\ \beta \end{array}$$

where each α_i and β has the form $\vdash A$ ($\vdash$ denotes provability in classical logic) or $B \mid\!\sim C$, such that A, B, and C are propositional formulas. If $n = 0$, then the rule is called an axiom. $\alpha_1, \alpha_2, \ldots, \alpha_n$ form the *premise* of the property, whereas β is its conclusion. As an example, the property of LLE (left logical equivalence), characterizing all KLM logics, is denoted by the following rule:

$$\genfrac{}{}{0pt}{}{\vdash A \leftrightarrow B \qquad A \mid\!\sim C}{B \mid\!\sim C}$$

They distinguish between three different *kinds of properties* of a nonmonotonic (KLM) system:

- type 1: the monotonic deduction symbol $\vdash$ does not appear in the property;
- type 2: the monotonic deduction symbol $\vdash$ appears only into the premises of the property;
- type 3: the monotonic deduction symbol $\vdash$ appears into the conclusion of the property.

Properties of type 1 correspond to axioms of conditional logics, whereas properties of type 2 correspond to conditional inference rules. Properties of type 3 are negative inference rules that do not correspond to anything ever used in logic.

Crocco and Lamarre then introduce a systematic translation of expressions of nonmonotonic logics into expressions of conditional logics; this operation, denoted with the symbol "*", is defined as follows:

Definition 3.30 (Definition 2.6 in [CL92]). *Let F be an expression of a KLM logic. F^* will be the rewritten expression of F in the language of conditional logics obtained in the following way:*

- *for every classical expression F, $F^* = F$*
- *for every expression $F \equiv A \mid\!\sim B$, $F^* \equiv A \Rightarrow B$*
- *for every expression $F = \neg(A \mid\!\sim B)$, $F^* \equiv \neg(A \Rightarrow B)$*
- *for every property of type 1 of the form*

$$F \equiv \genfrac{}{}{0pt}{}{\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_n}{\beta},$$

$n \geq 0$, we have

$$F^* \equiv (\alpha_1^* \wedge \alpha_2^* \wedge \ldots \wedge \alpha_n^*) \rightarrow \beta^*$$

- *for every property of type 2 of the form*

$$F \equiv \frac{\vdash \alpha_1 \;\; \vdash \alpha_2 \quad \dots \quad \vdash \alpha_n \quad \beta_1\ \beta_2 \ldots \beta_m}{\gamma},$$

$n > 0$ and $m \geq 0$, we have

$$F^* \equiv \genfrac{}{}{0pt}{}{\vdash \alpha_1 \;\; \vdash \alpha_2 \ldots \vdash \alpha_n}{\vdash (\beta_1^* \wedge \beta_2^* \wedge \ldots \wedge \beta_m^*) \rightarrow \gamma^*}$$

- *for every property of type 3 of the form*

$$F \equiv \genfrac{}{}{0pt}{}{\vdash \alpha_1 \;\; \vdash \alpha_2 \ldots \vdash \alpha_n \quad \beta_1\ \beta_2 \ldots \beta_m}{\gamma},$$

REF. $A \mid\!\sim A$

LLE. $$\frac{\vdash A \leftrightarrow B \qquad A \mid\!\sim C}{B \mid\!\sim C}$$

RW. $$\frac{\vdash A \rightarrow B \qquad C \mid\!\sim A}{C \mid\!\sim B}$$

CM. $$\frac{A \mid\!\sim B \qquad A \mid\!\sim C}{A \wedge B \mid\!\sim C}$$

AND. $$\frac{A \mid\!\sim B \qquad A \mid\!\sim C}{A \mid\!\sim (B \wedge C)}$$

OR. $$\frac{A \mid\!\sim C \qquad B \mid\!\sim C}{A \vee B \mid\!\sim C}$$

RM. $$\frac{A \mid\!\sim B \qquad \neg(A \mid\!\sim \neg C)}{(A \wedge C) \mid\!\sim B}$$

Figure 3.7: Axioms and rules of **R** (Crocco and Lamarre's notation).

$n \geq 0$ and $m \geq 0$, we have

$$F^* \equiv \begin{array}{c} \vdash \alpha_1 \ \vdash \alpha_2 \dots \vdash \alpha_n \quad \not\vdash \gamma \\ \vdash \neg(\beta_1^* \wedge \beta_2^* \wedge \dots \wedge \beta_m^*) \end{array}$$

By the above translation, they prove the following general representation theorem:

Theorem 3.31 (Corollary 2.1 in [CL92]). *Let $\mathcal{P}_{1,2}$ be a set of properties of type 1 or 2, defining a class of nonmonotonic relations C, and containing all classical tautologies. Let $\mathcal{P}^*_{1,2}$ be the logical system containing all the axioms of classical logic, closed under classical inference rules and under the translation by the operation "*" of the nonmonotonic properties of $\mathcal{P}_{1,2}$.*
$\mid\!\sim$ is a deduction relation verifying all properties of $\mathcal{P}_{1,2}$ if and only if there is an interpretation $\mathcal{M}$ of $\mathcal{P}^*_{1,2}$ such that $A \mid\!\sim B$ iff $\mathcal{M} \models A \Rightarrow B$.

By Theorem 3.31 it is easy to prove that the KLM logics correspond to the flat fragment of well known conditional logics. Let us analyze this in detail, by considering each logic of the KLM framework.

Rational Logic R

First, we use Crocco and Lamarre's notation to denote the properties of the logic **R**. The resulting axiom system is shown in Figure 3.7.
Applying the rewriting operation of Definition 3.30, we obtain the set of axioms and rules of Figure 3.8.

$$A \Rightarrow A \qquad \text{(ID)}$$

$$\frac{\vdash A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)} \qquad \text{(RCEA)}$$

$$\frac{\vdash A \rightarrow B}{(C \Rightarrow A) \rightarrow (C \Rightarrow B)} \qquad \text{(RCK)}$$

$$((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow ((A \wedge B) \Rightarrow C) \qquad \text{(AC)}$$

$$((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \Rightarrow (B \wedge C))$$

$$((A \Rightarrow C) \wedge (B \Rightarrow C)) \rightarrow ((A \vee B) \Rightarrow C) \qquad \text{(CA)}$$

$$((A \Rightarrow B) \wedge \neg(A \Rightarrow \neg C)) \rightarrow ((A \wedge C) \Rightarrow B) \qquad \text{(CV)}$$

Figure 3.8: Axioms and rules of **R** after rewriting.

In [Nut80] it is shown that this set of rules and axioms, added to any axiomatization of classical logic, gives an equivalent axiomatization of the conditional logic CK+ID+AC+CUT+CA+CV, also known as **V** [Lew73] or **CE+CV**. The selection function semantics associated with this logic is given by considering a selection function f having the following properties:

- $f(w, [A]) \subseteq [A]$
- if $f(w, [A]) \subseteq [B]$ then $f(w, [A]) = f(w, [A \wedge B])$
- if $f(w, [A]) \subseteq [B]$ then $f(w, [A \wedge B]) \subseteq f(w, [A])$
- $f(w, [A \wedge B]) \subseteq f(w, [A]) \cup f(w, [B])$
- if $f(w, [A]) \subseteq [B]$ and $f(w, [A]) \cap [C] \neq \emptyset$ then $f(w, [A \wedge C]) \subseteq [B]$.

Preferential Logic P

Since the properties of **P** can be obtained by the properties of **R** by removing RM, it immediately follows that the corresponding system of conditional logics is CK+ID+AC+CUT+CA, also known as **CE** [GGOS03]. This correspondence has been already presented in [KLM90]. This logic is also known as the conditional system **WC§**, corresponding to the logic **WC** defined in [Nut80] without the axiom MP.

Loop-Cumulative Logic CL

The axiomatization of **CL** is obtained from the axiomatization of **P** by removing OR and by adding LOOP and CUT. Therefore, the axiom system obtained by the translation of "*" does not contain (CA), whereas it contains the following set of axioms:

- $((A_0 \Rightarrow A_1) \wedge \ldots \wedge (A_{n-1} \Rightarrow A_n) \wedge (A_n \Rightarrow A_0)) \rightarrow (A_0 \Rightarrow A_n)$

- $((A \Rightarrow B) \wedge ((A \wedge B) \Rightarrow C)) \rightarrow (A \Rightarrow C)$ (CUT)

As observed in [KLM90], **CL** is the only KLM logic having no correspondence with any known conditional logic.

Cumulative Logic C

The axiom system of **C** is obtained from **CL**'s by removing LOOP. Therefore, **C** corresponds to the flat fragment of conditional logic CK+ID+AC+CUT. In [AGR02], the authors call this conditional logic **CU**. Notice that the restriction on the selection function f corresponding to the axiom (AC) is as follows:

- if $f(w, [A]) \subseteq [B]$ then $f(w, [A \wedge B]) \subseteq f(w, [A])$

Chapter 4

A Sequent Calculus for Standard Conditional Logics

In this chapter we briefly recall the conditional logics introduced in Chapter 2; in particular, we focus on the selection function semantics, then on the corresponding basic system, CK, and its extensions with axioms ID, MP, CS, and CEM. We present a cut-free sequent calculus, called SeqS, for these conditional logics. The calculus uses labels and transition formulas and can be used to prove decidability and space complexity bounds for the respective logics. We also show that these calculi can be the base for uniform proof systems.

The main results presented in this chapter can also be found in [OPS07, OP08].

4.1 Introduction

In Chapter 2 we have introduced conditional logics, and we have observed that, in spite of their significance, very few proof systems have been proposed for conditional logics. We have also observed that one possible reason of the underdevelopment of proof-methods for conditional logics is the lack of a universally accepted semantics for them. This is in sharp contrast to modal and temporal logics which have a consolidated semantics based on a standard kind of Kripke structures.

In this chapter we focus on the most general solution of the *selection function semantics*. With the selection function semantics, truth values are assigned to formulas depending on a world; intuitively, the selection function f selects, for a world w and a formula A, the set of worlds $f(w, A)$ which are "most-similar to w" or "closer to w" given the information A. In *normal* conditional logics, the function f depends on the set of worlds satisfying A rather than on A itself, so that $f(w, A) = f(w, A')$ whenever A and A' are true in the same worlds (normality condition). A conditional sentence $A \Rightarrow B$ is true in w whenever B is true in every world selected by f for A and w. It is the normality condition which marks essentially the difference between conditional logics on the one hand, and multimodal logic, on the other (where one might well have a family of $\Box$ indexed by formulas). We believe that it is the very condition of normality what makes difficult to develop proof systems for conditional logics with the selection function semantics.

Since we adopt the selection function semantics, CK is the fundamental system; it has the same role as the system K (from which it derives its name) in modal logic: CK-valid formulas are exactly those ones that are valid in every selection function model.

In this chapter we present a sequent calculus for CK and for some of its standard extensions, namely CK+{ID, MP, CS, CEM} including most of the combinations of these extensions. To the best of our knowledge, the presented calculi are the first ones for these logics. Our calculi make use of labels, following the line of [Vig00] and [Gab96]. Two types of formulas are involved in the rules of the calculi: world formulas of the form $x : A$ representing that A holds at world x and transition formulas of the form $x \xrightarrow{A} y$ representing that $y \in f(x, A)$. The rules manipulate both kinds of formulas.

We are able to give cut-free calculi for CK and all its extensions in the set {ID, MP, CS, CEM}, except those including *both* CEM and MP. The completeness of the calculi is an immediate consequence of the admissibility of cut.

We show that one can derive a decision procedure from the cut-free calculi. Whereas the decidability of these systems was already proved by Nute (by a finite-model property argument), our calculi give the first *constructive* proof of decidability. As usual, we obtain a terminating proof search mechanism by controlling the backward application of some critical rules. By estimating the size of the finite derivations of a given sequent, we also obtain a polynomial space complexity bound for these logics. We can also obtain a tighter complexity bound for the logics CK{+ID}, as they satisfy a kind of *disjunction property.*

Our calculi can be the starting point to develop goal-oriented proof procedures, according to the paradigm of Uniform Proofs by Miller and others [MNPS91, GO00]. Calculi of these kind are suitable for logic programming applications. As a preliminary result, we present a goal-directed calculus called $\mathcal{U}$S', for a fragment of CK and its extensions with ID and MP, where the "clauses" are a sort of conditional Harrop formulas.

4.2 Recall to Conditional Logics

In this section we briefly recall conditional logics, introduced in Chapter 2. Conditional logics are extensions of classical logic obtained by adding the conditional operator $\Rightarrow$. In this chapter, we only consider propositional conditional logics.
A propositional conditional language $\mathcal{L}$ contains the following items:

- a set of propositional variables ATM;
- the symbol of *false* $\bot$;
- a set of connectives[1] $\rightarrow$, $\Rightarrow$.

We define formulas of $\mathcal{L}$ as follows:

- $\bot$ and the propositional variables of ATM are *atomic formulas*;

[1] The usual connectives $\top$, $\wedge$, $\vee$ and $\neg$ can be defined in terms of $\bot$ and $\rightarrow$.

- if A and B are formulas, $A \rightarrow B$ and $A \Rightarrow B$ are *complex formulas.*

We adopt the *selection function semantics*, that we briefly recall here. We consider a non-empty set of possible worlds $\mathcal{W}$. Intuitively, the selection function f selects, for a world w and a formula A, the set of worlds of $\mathcal{W}$ which are *closer* to w given the information A. A conditional formula $A \Rightarrow B$ holds in a world w if the formula B holds in *all the worlds selected by f for w and A.*

A model is a triple $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ where:

- $\mathcal{W}$ is a non empty set of items called *worlds*;
- f is the so-called *selection function* and has the following type:

$$f\colon \mathcal{W} \times 2^{\mathcal{W}} \longrightarrow 2^{\mathcal{W}}$$

- $[\,]$ is the *evaluation function*, which assigns to an atom $P \in ATM$ the set of worlds where P is true, and is extended to the other formulas as follows:

  $\star$ $[\bot] = \emptyset$

  $\star$ $[A \rightarrow B]$=($\mathcal{W}$ - $[A]$) $\cup$ $[B]$

  $\star$ $[A \Rightarrow B]$=$\{w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}$

We have defined f taking $[A]$ rather than A (i.e. $f(w,[A])$ rather than $f(w,A)$) as an argument; this is equivalent to define f on formulas, i.e. $f(w,A)$ but imposing that if $[A]$=$[A^{'}]$ in the model, then $f(w, A)$=$f(w, A^{'})$. This condition is called *normality.*

The semantics above characterizes the *basic conditional system*, called CK. An axiomatization of the CK system, alternative to the one presented in Chapter 2, is given by:

- any axiomatization of the classical propositional calculus;
- (Modus Ponens) $\dfrac{A \quad A \rightarrow B}{B}$
- (RCEA) $\dfrac{A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$
- (RCK) $\dfrac{(A_1 \wedge \cdots \wedge A_n) \rightarrow B}{(C \Rightarrow A_1 \wedge \cdots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)}$

Other conditional systems are obtained by assuming further properties on the selection function; we consider the following standard extensions of the basic system CK:

System	**Axiom**	**Model condition**
ID	$A \Rightarrow A$	$f(w, [A]) \subseteq [A]$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	$w \in [A] \rightarrow w \in f(w, [A])$
CS	$(A \wedge B) \rightarrow (A \Rightarrow B)$	$w \in [A] \rightarrow f(w, [A]) \subseteq \{w\}$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w, [A]) \mid \leq 1$

From now on we use the following notation: AX is the set of axioms considered, i.e. AX={CEM, CS, ID, MP}. S stands for any subset of AX, i.e. S $\subseteq$ AX. We also denote with S* each S' such that S $\subseteq$ S' $\subseteq$ AX; for instance, CS+ID* is used to represent any of the following systems: CK+CS+ID, CK+CEM+CS+ID, CK+CS+ID+MP and CK+CEM+CS+ID+MP.

The above axiomatization is complete with respect to the semantics (see Theorem 2.2 in Chapter 2). Observe that the condition CS is derivable in systems characterized by conditions CEM and MP. Indeed, for CEM we have that $(*)$ $\mid f(w, [A]) \mid \leq 1$; if $w \in [A]$, then we have that $w \in f(w, [A])$ by MP, but by $(*)$ we have that $f(w, [A]) = \{w\}$, satisfying the CS condition.

4.3 A Sequent Calculus for Conditional Logics

In this section we present **SeqS**, a sequent calculs for the conditional systems introduced above. The calculi make use of labels to represent possible worlds. We consider a language $\mathcal{L}$ and a denumerable alphabet of labels $\mathcal{A}$, whose elements are denoted by *x, y, z, ...* There are two kinds of labelled formulas:

1. *world formulas*, denoted by *x: A*, where $x \in \mathcal{A}$ and $A \in \mathcal{L}$, used to represent that A holds in a world x;

2. *transition formulas*, denoted by $x \xrightarrow{A} y$, where $x, y \in \mathcal{A}$ and $A \in \mathcal{L}$. A transition formula $x \xrightarrow{A} y$ represents that $y \in f(x, [A])$.

A *sequent* is a pair $\langle \Gamma, \Delta \rangle$, usually denoted with $\Gamma \vdash \Delta$, where Γ and Δ are multisets of labelled formulas.

For technical reasons we introduce the notion of positive/negative occurrences of a world formula:

Definition 4.1 (Positive and negative occurrences of a world formula)**.** *Given a world formula $x : A$, we say that:*

- *$x : A$ occurs positively in $x : A$;*

- *if a world formula $x : B \rightarrow C$ occurs positively (negatively) in $x : A$, then $x : C$ occurs positively (negatively) in $x : A$ and $x : B$ occurs negatively (positively) in $x : A$;*
- *if a formula $x : B \Rightarrow C$ occurs positively (negatively) in $x : A$, then $x : C$ occurs positively (negatively) in $x : A$.*

A world formula $x : A$ occurs positively (negatively) in a multiset Γ if $x : A$ occurs positively (negatively) in some world formula $x : G \in \Gamma$. Given $\Gamma \vdash \Delta$, we say that $x : A$ occurs positively (negatively) in $\Gamma \vdash \Delta$ if $x : A$ occurs positively (negatively) in Γ or $x : A$ occurs negatively (positively) in Δ.

The intuitive meaning of a sequent $\Gamma \vdash \Delta$ is: every model that satisfies all labelled formulas of Γ in the respective worlds (specified by the labels) satisfies at least one of the labelled formulas of Δ (in those worlds). This is made precise by the notion of *validity* of a sequent given in the next definition:

Definition 4.2 (Sequent validity)**.** *Given a model*

$$\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$$

for $\mathcal{L}$, and a label alphabet $\mathcal{A}$, we consider any mapping

$$I : \mathcal{A} \rightarrow \mathcal{W}$$

Let F be a labelled formula, we define $\mathcal{M} \models_I F$ as follows:

- $\mathcal{M} \models_I x$*: A* iff $I(x) \in [A]$
- $\mathcal{M} \models_I x \xrightarrow{A} y$ iff $I(y) \in f(I(x), [A])$

We say that $\Gamma \vdash \Delta$ is valid *in $\mathcal{M}$ if for every mapping $I : \mathcal{A} \rightarrow \mathcal{W}$, if $\mathcal{M} \models_I F$ for every $F \in \Gamma$, then $\mathcal{M} \models_I G$ for some $G \in \Delta$. We say that $\Gamma \vdash \Delta$ is valid in a system (CK or one of its extensions) if it is valid in every $\mathcal{M}$ satisfying the specific conditions for that system (if any).*

In Figures 4.1 and 4.2 we present the calculi for CK and its mentioned extensions. Observe that the restrictions $x \neq y$ on (**CS**) and $y \neq z$ on (**CEM**) have a different nature than the restriction on ($\Rightarrow$ **R**). The former ones are needed to avoid a looping application of the rules, as the right premise would be identical to the conclusion modulo label substitution. The latter one is necessary to preserve the soundness of the calculus.

Example 4.3. *We show a derivation of an instance of the* (**ID**) *axiom ($P \in ATM$).*

$$\dfrac{\dfrac{x \xrightarrow{P} y, y : P \vdash y : P}{x \xrightarrow{P} y \vdash y : P}\,(\mathbf{ID})}{\vdash x : P \Rightarrow P}\,(\Rightarrow \mathbf{R})$$

Example 4.4. *We show a derivation of an instance of the* (**MP**) *axiom ($P, Q \in ATM$).*

$$\dfrac{\dfrac{\dfrac{\dfrac{x : P \Rightarrow Q, x : P \vdash x \xrightarrow{P} x, x : P, x : Q}{x : P \Rightarrow Q, x : P \vdash x \xrightarrow{P} x, x : Q}\,(\mathbf{MP}) \qquad x : P \Rightarrow Q, x : P, x : Q \vdash x : Q}{x : P \Rightarrow Q, x : P \vdash x : Q}\,(\Rightarrow \mathbf{L})}{x : P \Rightarrow Q \vdash x : P \rightarrow Q}\,(\rightarrow \mathbf{R})}{\vdash x : (P \Rightarrow Q) \rightarrow (P \rightarrow Q)}\,(\rightarrow \mathbf{R})$$

$$(\mathbf{AX}) \quad \Gamma, x : P \vdash \Delta, x : P \quad (P \in ATM) \qquad\qquad (\mathbf{A}\bot) \quad \Gamma, x : \bot \vdash \Delta$$

$$(\rightarrow \mathbf{L}) \frac{\Gamma \vdash \Delta, x : A \qquad \Gamma, x : B \vdash \Delta}{\Gamma, x : A \rightarrow B \vdash \Delta} \qquad\qquad (\rightarrow \mathbf{R}) \ \frac{\Gamma, x : A \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : A \rightarrow B}$$

$$(\Rightarrow \mathbf{L}) \frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta}$$

$$(\Rightarrow \mathbf{R}) \ \frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{\Gamma \vdash \Delta, x : A \Rightarrow B} \ (y \notin \Gamma \cup \Delta) \qquad\qquad (\mathbf{EQ}) \ \frac{u : A \vdash u : B \qquad u : B \vdash u : A}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{B} y}$$

Figure 4.1: Sequent calculus SeqCK.

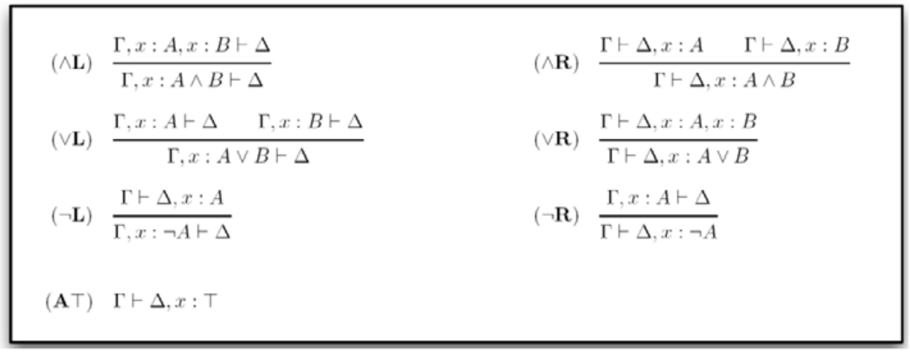
$$(\mathbf{ID}) \quad \begin{array}{c} \Gamma, x \xrightarrow{A} y, y : A \vdash \Delta \\ \Gamma, x \xrightarrow{A} y \vdash \Delta \end{array}$$

$$(\mathbf{MP}) \quad \begin{array}{c} \Gamma \vdash x \xrightarrow{A} x, x : A, \Delta \\ \Gamma \vdash x \xrightarrow{A} x, \Delta \end{array}$$

$$(\mathbf{CS}) \quad \frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A \qquad \Gamma[x/u, y/u], u \xrightarrow{A} u \vdash \Delta[x/u, y/u]}{\Gamma, x \xrightarrow{A} y \vdash \Delta} \ (x \neq y, u \notin \Gamma, \Delta)$$

$$(\mathbf{CEM}) \quad \frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z \qquad (\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]}{\Gamma, x \xrightarrow{A} y \vdash \Delta} \ (y \neq z, u \notin \Gamma, \Delta)$$

Figure 4.2: SeqS's rules for systems extending CK. We denote with $\Sigma[x/u]$ the multiset obtained from Σ by replacing the label x by u wherever it occurs.



$$(\wedge\mathbf{L})\ \frac{\Gamma, x : A, x : B \vdash \Delta}{\Gamma, x : A \wedge B \vdash \Delta} \qquad (\wedge\mathbf{R})\ \frac{\Gamma \vdash \Delta, x : A \qquad \Gamma \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : A \wedge B}$$

$$(\vee\mathbf{L})\ \frac{\Gamma, x : A \vdash \Delta \qquad \Gamma, x : B \vdash \Delta}{\Gamma, x : A \vee B \vdash \Delta} \qquad (\vee\mathbf{R})\ \frac{\Gamma \vdash \Delta, x : A, x : B}{\Gamma \vdash \Delta, x : A \vee B}$$

$$(\neg\mathbf{L})\ \frac{\Gamma \vdash \Delta, x : A}{\Gamma, x : \neg A \vdash \Delta} \qquad (\neg\mathbf{R})\ \frac{\Gamma, x : A \vdash \Delta}{\Gamma \vdash \Delta, x : \neg A}$$

$$(\mathbf{A}\top)\ \Gamma \vdash \Delta, x : \top$$

Figure 4.3: Additional axioms and rules in SeqS for the other boolean operators, derived from the rules in Figure 4.1 by the usual equivalences.

Example 4.5. *We show a derivation of an instance of the* (**CS**) *axiom (*$P, Q \in$ *ATM).*

$$\begin{array}{cl}
\dfrac{x : P, x : Q, x \xrightarrow{P} y \vdash y : Q, x : P \qquad u : P, u : Q, u \xrightarrow{P} u \vdash u : Q}{x : P, x : Q, x \xrightarrow{P} y \vdash y : Q} & (\mathbf{CS}) \\
 & (\wedge\mathbf{L}) \\
x : P \wedge Q, x \xrightarrow{P} y \vdash y : Q & \\
 & (\Rightarrow \mathbf{R}) \\
x : P \wedge Q \vdash x : P \Rightarrow Q & \\
 & (\rightarrow \mathbf{R}) \\
\vdash x : (P \wedge Q) \rightarrow (P \Rightarrow Q) &
\end{array}$$

Example 4.6. *We show a derivation of an instance of the* (**CEM**) *axiom (*$P, Q \in$ *ATM).*

$$\begin{array}{cl}
u : P \vdash u : P \qquad u : P \vdash u : P & \\
\qquad\qquad\qquad\qquad (\mathbf{EQ}) & \\
\dfrac{x \xrightarrow{P} y, x \xrightarrow{P} z, z : Q \vdash y : Q, x \xrightarrow{P} y \qquad x \xrightarrow{P} u, x \xrightarrow{P} u, u : Q \vdash u : Q}{x \xrightarrow{P} y, x \xrightarrow{P} z, z : Q \vdash y : Q} & (\mathbf{CEM}) \\
 & (\neg\mathbf{R}) \\
x \xrightarrow{P} y, x \xrightarrow{P} z \vdash y : Q, z : \neg Q & \\
 & (\Rightarrow \mathbf{R}) \\
x \xrightarrow{P} y \vdash y : Q, x : P \Rightarrow \neg Q & \\
 & (\Rightarrow \mathbf{R}) \\
\vdash x : P \Rightarrow Q, x : P \Rightarrow \neg Q & \\
 & (\vee\mathbf{R}) \\
\vdash x : (P \Rightarrow Q) \vee (P \Rightarrow \neg Q) &
\end{array}$$

4.3.1 Basic Structural Properties of SeqS

In order to prove that the sequent calculus SeqS is sound and complete with respect to the semantics, we introduce some structural properties.

First of all, we define the complexity of a labelled formula:

Definition 4.7 (Complexity of a labelled formula cp(F))**.** *We define the complexity of a labelled formula F as follows:*

1. *cp* $(x : A) = 2 * \mid A \mid$

2. $cp\ (x \xrightarrow{A} y) = 2 * \mid A \mid +1$

where $\mid A \mid$ *is the number of symbols occurring in the string representing the formula* A*.*

We can prove the following Lemma:

Lemma 4.8. *Given any multiset of formulas* Γ *and* Δ*, and a labelled formula* F*, we have that* $\Gamma, F \vdash \Delta, F$ *is derivable in SeqS.*

Proof. By induction on the complexity of the formula F. The proof is easy and left to the reader.

Lemma 4.9 (Height-preserving label substitution)**.** *If a sequent* $\Gamma \vdash \Delta$ *has a derivation of height* h*, then* $\Gamma[x/y] \vdash \Delta[x/y]$ *has a derivation of height* $\leq h$*, where* $\Gamma[x/y] \vdash \Delta[x/y]$ *is the sequent obtained from* $\Gamma \vdash \Delta$ *by replacing a label* x *by a label* y *wherever it occurs.*

Proof. By induction on the height of a derivation of $\Gamma \vdash \Delta$. We show the most interesting cases, the other ones are easy and left to the reader. We begin with the case when (**CS**) is applied to $\Gamma \vdash \Delta$ with a derivation of height h of the form:

$$\dfrac{(1)\Gamma', y \xrightarrow{A} x \vdash y : A, \Delta \qquad (2)\Gamma'[y/u, x/u], u \xrightarrow{A} u \vdash \Delta[y/u, x/u]}{\Gamma', y \xrightarrow{A} x \vdash \Delta}\ (\mathbf{CS})$$

Our goal is to find a proof of height $\leq h$ of $\Gamma'[x/y], y \xrightarrow{A} y \vdash \Delta[x/y]$. Applying the inductive hypothesis to (2), we have a proof of height $\leq h-1$ of the sequent $(3)(\Gamma'[y/u, x/u])[u/y], y \xrightarrow{A} y \vdash (\Delta[y/u, x/u])[u/y]$, but (3) corresponds to $\Gamma'[x/y], y \xrightarrow{A} y \vdash \Delta[x/y]$, since labels x and y have both been replaced by a *new* label u in (2), and the proof is over.

The other interesting case is when ($\Rightarrow$ **R**) is the rule ending the derivation (i.e. the rule applied to $\Gamma \vdash \Delta$); the situation is as follows:

$$\dfrac{\Gamma, x \xrightarrow{A} y \vdash \Delta', y : B}{\Gamma \vdash \Delta', x : A \Rightarrow B}\ (\Rightarrow \mathbf{R})$$

We show that there is a derivation of $\Gamma[x/y] \vdash \Delta'[x/y], y : A \Rightarrow B$, of height less or equal than h. First, observe that the label y in the above derivation is a *new* label, not occurring in the conclusion of ($\Rightarrow$ **R**); therefore, we can rename it with another new label, for instance w:

$$\dfrac{\Gamma, x \xrightarrow{A} w \vdash \Delta', w : B}{\Gamma \vdash \Delta', x : A \Rightarrow B}\ (\Rightarrow \mathbf{R})$$

Applying the inductive hypothesis on the premise of ($\Rightarrow$ **R**), we obtain a proof of $\Gamma[x/y], y \xrightarrow{A} w \vdash \Delta'[x/y], w : B$ of height no greater than $h-1$. We conclude by an application of ($\Rightarrow$ **R**), obtaining a proof (height $\leq h$) of $\Gamma[x/y] \vdash \Delta'[x/y], y : A \Rightarrow B$.

■

Theorem 4.10 (Height-preserving admissibility of weakening)**.** *If a sequent $\Gamma \vdash \Delta$ has a derivation of height h, then $\Gamma \vdash \Delta, F$ and $\Gamma, F \vdash \Delta$ have a derivation of height $\leq h$.*

Proof. By induction on the height of a derivation of $\Gamma \vdash \Delta$. The proof is easy and left to the reader.

Theorem 4.11 (Height-preserving invertibility of rules)**.** *Let $\Gamma \vdash \Delta$ be the conclusion of an application of one of the SeqS's rules, say R, with R different from* (**EQ**)*. If $\Gamma \vdash \Delta$ is derivable, then the premise(s) of R is (are) derivable with a derivation of (at most) the same height, i.e. SeqS's rules are height-preserving invertible.*

Proof. We consider each of the rules. We distinguish two groups of rules:

(*i*) ($\rightarrow$ **L**), ($\rightarrow$ **R**), and ($\Rightarrow$ **R**): for these rules we proceed by an inductive argument on the height of a proof of their conclusions; the cases of ($\rightarrow$ **L**) and ($\rightarrow$ **R**) are easy and left to the reader. The proof for ($\Rightarrow$ **R**) is as follows: for any y, if $\Gamma \vdash \Delta, x : A \Rightarrow B$ is an axiom, then $\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B$ is an axiom too, since axioms are restricted to atomic formulas. If $h > 0$ and the proof of $\Gamma \vdash \Delta, x : A \Rightarrow B$ is concluded (looking forward) by any rule other than ($\Rightarrow$ **R**), we apply the inductive hypothesis to the premise(s), then we conclude by applying the same rule. If the derivation of $\Gamma \vdash \Delta, x : A \Rightarrow B$ is ended by ($\Rightarrow$ **R**) we have the following subcases:

⋆ $x : A \Rightarrow B$ is the principal formula of ($\Rightarrow$ **R**): the proof is ended as follows:

$$\dfrac{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{\Gamma \vdash \Delta, x : A \Rightarrow B}\ (\Rightarrow \mathbf{R})$$

We have a proof of $\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B$ of height $h - 1$ and the proof is over;

⋆ $x : A \Rightarrow B$ is not the principal formula of ($\Rightarrow$ **R**): the proof is ended as follows:

$$\dfrac{\Gamma, w \xrightarrow{C} z \vdash \Delta, z : D, x : A \Rightarrow B}{\Gamma \vdash \Delta, x : A \Rightarrow B, w : C \Rightarrow D}\ (\Rightarrow \mathbf{R})$$

where z is a "new" label and then, without loss of generality, we can assume that z is not y, since we can apply the height-preserving label substitution. By inductive hypothesis on the premise we obtain a derivation of $\Gamma, w \xrightarrow{C} z, x \xrightarrow{A} y \vdash \Delta, z : D, y : B$ from which we conclude as follows:

$$\dfrac{\Gamma, w \xrightarrow{C} z, x \xrightarrow{A} y \vdash \Delta, z : D, y : B}{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B, w : C \Rightarrow D}\ (\Rightarrow \mathbf{R})$$

(*ii*) ($\Rightarrow$ **L**), (**ID**), (**MP**), (**CS**), and (**CEM**): these rules are height-preserving invertible, since their premise(s) is (are) obtained by weakening from the conclusion, and weakening is height-preserving admissible (Theorem 4.10). We consider each of the rules:

- ($\Rightarrow$ **L**): its premises are obtained by weakening from the conclusion, i.e. given a proof (height h) of $\Gamma, x : A \Rightarrow B \vdash \Delta$, by weakening we have proofs of height $\leq h$ of $\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y$ and $\Gamma, x : A \Rightarrow B, y : B \vdash \Delta$;
- (**CS**): given a derivation of $\Gamma, x \xrightarrow{A} y \vdash \Delta$, we can obtain a proof of no greater height of $\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A$ since weakening is height-preserving admissible; we can also obtain a proof of $\Gamma[x/u, y/u], u \xrightarrow{A} u \vdash \Delta[x/u, y/u]$ since label substitution is height-preserving admissible (Lemma 4.9);
- (**CEM**): given a proof (height h) of $\Gamma, x \xrightarrow{A} y \vdash \Delta$, we can obtain a proof of at most the same height of $\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z$ by weakening. There is also a proof (height $\leq h$) of $(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]$ by the height-preserving label substitution.

The cases of (**ID**) and (**MP**) are similar and left to the reader.

■

It is worth noticing that the height-preserving invertibility also preserves the number of applications of the rules in a proof, that is to say: if $\Gamma_1 \vdash \Delta_1$ is derivable by Theorem 4.11 since it is the premise of a backward application of an invertible rule R to $\Gamma_2 \vdash \Delta_2$, then it has a derivation containing *the same rule applications* of the proof of $\Gamma_2 \vdash \Delta_2$. For instance, if (1) $\Gamma, x \xrightarrow{A} y \vdash \Delta$ is derivable with a proof Π, then (2) $\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta$ is derivable since (**ID**) is invertible; moreover, there exists a proof of (2) containing the same rules of Π, obtained by adding $y : A$ in each sequent of Π from which (1) descends. This fact will be systematically used throughout the chapter, in the sense that we will assume that every proof transformation due to the invertibility preserves the number of rules applications in the initial proof.

Theorem 4.12 (Height-preserving admissibility of contraction)**.** *The rules of contraction are height-preserving admissible in SeqS, i.e. if a sequent* $\Gamma \vdash \Delta, F, F$ *is derivable in SeqS, then there is a derivation of no greater height of* $\Gamma \vdash \Delta, F$*, and if a sequent* $\Gamma, F, F \vdash \Delta$ *is derivable in SeqS, then there is a derivation of no greater height of* $\Gamma, F \vdash \Delta$*. Moreover, the proof of the contracted sequent does not add any rule application to the initial proof* [2]*.*

Proof. By simultaneous induction on the height of derivation for left and right contraction. If $h = 0$, i.e. $\Gamma \vdash \Delta, F, F$ is an axiom, then we have to consider the following subcases:

- $w : \bot \in \Gamma$: in this case, obviously $\Gamma \vdash \Delta, F$ is an axiom too;
- an atom $G \in \Gamma \cap \Delta$: we conclude, since $\Gamma \vdash \Delta, F$ is an axiom too;
- F is an atom and $F \in \Gamma$: the proof is over, observing that $\Gamma \vdash \Delta, F$ is an axiom too.

The proof of the case where $\Gamma, F, F \vdash \Delta$ is an axiom is symmetric.

If $h > 0$, consider the last rule applied (looking forward) to derive the premise of contraction. We distinguish two cases:

[2] In this case we say that contractions are rule-preserving admissible.

- the contracted formula F is not principal in it: in this case, both occurrences of F are in the premise(s) of the rule, which have a smaller derivation height. By the inductive hypothesis, they can be contracted and the conclusion is obtained by applying the rule to the contracted premise(s). As an example, consider a proof ended by an application of (**CS**) as follows:

$$\dfrac{\Gamma^{'}, x \xrightarrow{A} y, F, F \vdash \Delta, x : A \quad \Gamma^{'}[x/u, y/u], u \xrightarrow{A} u, F[x/u, y/u], F[x/u, y/u] \vdash \Delta[x/u, y/u]}{\Gamma^{'}, x \xrightarrow{A} y, F, F \vdash \Delta}\ (\mathbf{CS})$$

  By the inductive hypothesis, we have a proof of no greater height than the respective premise of the sequents $\Gamma^{'}, x \xrightarrow{A} y, F \vdash \Delta, x : A$ and $\Gamma^{'}[x/u, y/u], u \xrightarrow{A} u, F[x/u, y/u] \vdash \Delta[x/u, y/u]$, from which we conclude as follows, obtaining a proof of (at most) the same height as the initial proof:

$$\dfrac{\Gamma^{'}, x \xrightarrow{A} y, F \vdash \Delta, x : A \qquad \Gamma^{'}[x/u, y/u], u \xrightarrow{A} u, F[x/u, y/u] \vdash \Delta[x/u, y/u]}{\Gamma^{'}, x \xrightarrow{A} y, F \vdash \Delta}\ (\mathbf{CS})$$

- the contracted formula F is principal in it: we consider all the rules:

  - ($\rightarrow$ **L**): the proof is ended as follows:

$$\dfrac{(1)\Gamma, x : A \rightarrow B \vdash \Delta, x : A \qquad (2)\Gamma, x : A \rightarrow B, x : B \vdash \Delta}{\Gamma, x : A \rightarrow B, x : A \rightarrow B \vdash \Delta}\ (\rightarrow \mathbf{L})$$

    Since ($\rightarrow$ **L**) is height-preserving invertible (see Theorem 4.11), there is a derivation of no greater height than (1) of $(1a)\Gamma \vdash \Delta, x : A, x : A$ and $(1b)\Gamma, x : B \vdash \Delta, x : A$ and of no greater height than (2) of $(2a)\Gamma, x : B \vdash \Delta, x : A$ and $(2b)\Gamma, x : B, x : B \vdash \Delta$. Applying the inductive hypothesis on $(1a)$ and $(2b)$ and applying ($\rightarrow$ **L**) to the contracted sequents, we obtain a derivation of no greater height ending with (be $(1a^{'})$ and $(2b^{'})$ the contracted sequents):

$$\dfrac{(1a^{'})\Gamma \vdash \Delta, x : A \qquad (2b^{'})\Gamma, x : B \vdash \Delta}{\Gamma, x : A \rightarrow B \vdash \Delta}\ (\rightarrow \mathbf{L})$$

  - ($\rightarrow$ **R**): we proceed as in the previous case, since ($\rightarrow$ **R**) is height-preserving invertible;
  - ($\Rightarrow$ **L**): we have a proof ending with:

$$\dfrac{\Gamma, x : A \Rightarrow B, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \quad \Gamma, x : A \Rightarrow B, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow \mathbf{L})$$

    Applying the inductive hypothesis to both premises we can immediately conclude as follows:

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow \mathbf{L})$$

- $(\Rightarrow \mathbf{R})$: the proof is ended by:

$$\begin{array}{c}\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A \Rightarrow B, y : B \\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : A \Rightarrow B\end{array}\ (\Rightarrow \mathbf{R})$$

Applying the height-preserving invertibility of $(\Rightarrow \mathbf{R})$, we have a proof of $(1)\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} z \vdash \Delta, y : B, z : B$. Applying the height-preserving label substitution (Lemma 4.9) to (1), replacing z with y, we obtain a derivation of the sequent (2) $\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta, y : B, y : B$, since y and z are new labels not occurring in Γ, Δ. We can then apply the inductive hypothesis on (2), obtaining a proof of (3) $\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B$, from which we conclude by an application of $(\Rightarrow \mathbf{R})$:

$$\begin{array}{c}(3)\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B \\ \Gamma \vdash \Delta, x : A \Rightarrow B\end{array}\ (\Rightarrow \mathbf{R})$$

- $(\mathbf{EQ})$: the proof is ended as follows:

$$\begin{array}{c}u : A \vdash u : A^{'} \qquad u : A^{'} \vdash u : A \\ \Gamma^{'}, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A^{'}} y\end{array}\ (\mathbf{EQ})$$

It is easy to observe that $(\mathbf{EQ})$ does not require the second occurrence of $x \xrightarrow{A} y$; thus we obtain the following proof:

$$\begin{array}{c}u : A \vdash u : A^{'} \qquad u : A^{'} \vdash u : A \\ \Gamma^{'}, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A^{'}} y\end{array}\ (\mathbf{EQ})$$

The other half is symmetric ($\Gamma, x \xrightarrow{A^{'}} y \vdash \Delta^{'}, x \xrightarrow{A} y, x \xrightarrow{A} y$ derives from $(\mathbf{EQ})$);

- $(\mathbf{ID})$: given a proof ending with:

$$\begin{array}{c}\Gamma, y : A, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta \\ \Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta\end{array}\ (\mathbf{ID})$$

we can apply the inductive hypothesis on the premise, obtaining a proof of no greater height of $\Gamma, y : A, x \xrightarrow{A} y \vdash \Delta$, from which we conclude by an application of $(\mathbf{ID})$;

- $(\mathbf{MP})$: the proof is ended as follows:

$$\frac{\Gamma \vdash \Delta, x \xrightarrow{A} x, x \xrightarrow{A} x, x : A}{\Gamma \vdash \Delta, x \xrightarrow{A} x, x \xrightarrow{A} x} \, (\mathbf{MP})$$

We can apply the inductive hypothesis on the premise, obtaining a proof of no greater height of $\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A$, from which we conclude by an application of (**MP**);

- (**CEM**): given a proof ending with:

$$\frac{\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z \qquad (\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]}{\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta} \, (\mathbf{CEM})$$

we can apply the inductive hypothesis on the two premises, obtaining proofs of $\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z$ and $(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]$, from which we conclude by an application of (**CEM**);

- (**CS**): given a derivation ending as follows:

$$\frac{\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta, x : A \qquad (\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta)[x/u, y/u]}{\Gamma, x \xrightarrow{A} y, x \xrightarrow{A} y \vdash \Delta} \, (\mathbf{CS})$$

by the inductive hypothesis on the premises we can find derivations of $\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A$ and $(\Gamma, x \xrightarrow{A} y \vdash \Delta)[x/u, y/u]$, then we conclude by an application of (**CS**).

Notice that, in each case, the proof is concluded by applying the same rule under consideration, i.e. the derivation of the contracted sequent does not add any rule application to the proof of the initial sequent.

■

We now consider the cut rule:

$$\frac{\Gamma \vdash \Delta, F \quad F, \Gamma \vdash \Delta}{\Gamma \vdash \Delta} \, (cut)$$

where F is any labelled formula. We prove that this rule is admissible in all SeqS calculi, except those which contain *both* CEM and MP. For the systems without CEM the proof follows the standard pattern. For systems with CEM it is more complicated. For systems with both CEM and MP the admissibility of cut is open at present: we are unable to prove it, and we do not have a counterexample. A broader discussion can be found in Appendix A in [OPS05] and at the end of the proof of Theorem 4.13 below.

Now we prove that cut is admissible in all SeqS systems, except for systems allowing both (**CEM**) and (**MP**). From now on, we restrict our concern to all other systems.

Theorem 4.13 (Admissibility of cut)**.** *In systems SeqS - SeqCEM+MP* if* $\Gamma \vdash \Delta, F$ *and* $F, \Gamma \vdash \Delta$ *are derivable, so* $\Gamma \vdash \Delta$.

Proof. As usual, the proof proceeds by a double induction over the complexity of the cut formula and the sum of the heights of the derivations of the two premises of cut, in the sense that we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations. We first consider the case of systems without (**CEM**). We have several cases: (i) one of the two premises is an axiom, (ii) the last step of *one* of the two premises is obtained by a rule in which F is *not* the principal formula, (iii) F is the principal formula in the last step of *both* derivations.

(i) If one of the two premises is an axiom then either $\Gamma \vdash \Delta$ is an axiom, or the premise which is not an axiom contains two copies of F and $\Gamma \vdash \Delta$ can be obtained by contraction, which is admissible (see Theorem 4.12 above).

(ii) We distinguish two cases:

1. the sequent where F is not principal is derived by any rule (**R**), except the (**EQ**) rule. This case is standard, we can permute (**R**) over the cut: i.e. we cut the premise(s) of (**R**) and then we apply (**R**) to the result of cut. As an example, consider the case when $F = x \xrightarrow{A} y$ and it is the principal formula of an application of (**CS**) in the right derivation, and (**CS**) is also the last rule in the left derivation; the situation is as follows (we denote the substitution $\Sigma[x/u, y/u]$ with $\Sigma(u)$):

$$\dfrac{\dfrac{\begin{array}{l}(1)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y, x : A^{'} \\ (2)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u), u \xrightarrow{A} u\end{array}}{(5)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y}\,(\mathbf{CS}) \qquad \dfrac{\begin{array}{l}(3)\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x : A \\ (4)\Gamma^{'}(u), u \xrightarrow{A^{'}} u, u \xrightarrow{A} u \vdash \Delta(u)\end{array}}{(6)\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CS})}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}\,(cut)$$

We can apply the inductive hypothesis on the height to replace the following cut[3]:

$$\dfrac{(2)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u), u \xrightarrow{A} u \qquad (4)\Gamma^{'}(u), u \xrightarrow{A^{'}} u, u \xrightarrow{A} u \vdash \Delta(u)}{(7)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u)}\,(cut)$$

We replace the initial cut as follows:

$$\dfrac{\dfrac{\begin{array}{l}(1)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A^{'}, x \xrightarrow{A} y \\ (6^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x : A^{'}\end{array}}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A^{'}}\,(cut) \qquad (7)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u)}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}\,(\mathbf{CS})$$

where $(6^{'})$ is obtained by weakening from (6).

[3] Notice that cutting (4) and (2) corresponds to cutting (2) and (6) with the necessary label substitutions, i.e. permuting (**CS**) over the cut.

2. if one of the sequents, say $\Gamma \vdash \Delta, F$ is obtained by the (**EQ**) rule, where F is not principal, then also $\Gamma \vdash \Delta$ is derivable by the (**EQ**) rule and we are done.

(iii) F is the principal formula in both the inferences steps leading to the two cut premises. There are seven subcases: F is introduced (a) by ($\rightarrow$ **L**), ($\rightarrow$ **R**), (b) by ($\Rightarrow$ **L**), ($\Rightarrow$ **R**), (c) by (**EQ**), (d) by (**ID**) on the left and by (**EQ**) on the right, (e) by (**EQ**) on the left and by (**MP**) on the right, (f) by (**ID**) on the left and by (**MP**) on the right and (g) by (**CS**) on the left and by (**EQ**) on the right. The list is exhaustive[4].

(a) This case is standard and left to the reader.

(b) $F = x : A \Rightarrow B$ is introduced by ($\Rightarrow$ **R**) and ($\Rightarrow$ **L**). Then we have

$$\dfrac{\dfrac{(1)\Gamma, x \xrightarrow{A} z \vdash \Delta, z : B}{(2)\Gamma \vdash \Delta, x : A \Rightarrow B}(\Rightarrow \mathbf{R}) \qquad \dfrac{(3)\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \quad (4)\Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{(5)\Gamma, x : A \Rightarrow B \vdash \Delta}(\Rightarrow \mathbf{L})}{\Gamma \vdash \Delta}(cut)$$

where z does not occur in Γ, Δ and $z \neq x$; By Lemma 4.9, we obtain that $(1^{'})$ $\Gamma, x \xrightarrow{A} y \vdash y : B, \Delta$ is derivable by a derivation of no greater height than (1); moreover, we can obtain a proof of no greater height of $(2^{'})$ $\Gamma \vdash \Delta, x : A \Rightarrow B, x \xrightarrow{A} y$ and of $(2^{''})$ $\Gamma, y : B \vdash \Delta, x : A \Rightarrow B$, both by weakening from (2) (Theorem 4.10).

First, we can make the following cut, which uses the inductive hypothesis on the height:

$$\dfrac{(2^{'})\Gamma \vdash \Delta, x : A \Rightarrow B, x \xrightarrow{A} y \qquad (3)\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y}{(6)\Gamma \vdash \Delta, x \xrightarrow{A} y}(cut)$$

By weakening, we have a proof of no greater height than (6) of $(6^{'})$ $\Gamma \vdash \Delta, x \xrightarrow{A} y, y : B$. Thus we can replace the initial cut as follows:

$$\dfrac{\dfrac{\begin{array}{c}(2^{''})\Gamma, y : B \vdash \Delta, x : A \Rightarrow B \\ (4)\Gamma, x : A \Rightarrow B, y : B \vdash \Delta\end{array}}{\Gamma, y : B \vdash \Delta}(cut) \qquad \dfrac{\begin{array}{c}(1^{'})\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B \\ (6^{'})\Gamma \vdash \Delta, x \xrightarrow{A} y, y : B\end{array}}{\Gamma \vdash \Delta, y : B}(cut)}{\Gamma \vdash \Delta}(cut)$$

The upper cut on the left uses the induction hypothesis on the height, the others the induction hypothesis on the complexity of the cut formula.

[4] Notice that the case when $F = x \xrightarrow{A} x$ is introduced by (**CS**) on the left and (**MP**) on the right has not to be considered, since (**CS**) introduces a transition $x \xrightarrow{A} y$ (looking forward) only if $x \neq y$.

(*c*) $F = x \xrightarrow{A} y$ is introduced by (**EQ**) in both premises, we have

$$\dfrac{\dfrac{(1)\, u : A^{'} \vdash u : A \;\; (2)\, u : A \vdash u : A^{'}}{\Gamma', x \xrightarrow{A^{'}} y \vdash x \xrightarrow{A} y, \Delta}\,(\mathbf{EQ}) \qquad \dfrac{(3)\, u : A \vdash u : A^{''} \;\; (4)\, u : A^{''} \vdash u : A}{\Gamma, x \xrightarrow{A} y \vdash x \xrightarrow{A^{''}} y, \Delta'}\,(\mathbf{EQ})}{\Gamma', x \xrightarrow{A^{'}} y \vdash x \xrightarrow{A^{''}} y, \Delta'}\,(cut)$$

where $\Gamma = \Gamma', x \xrightarrow{A^{'}} y$, $\Delta = x \xrightarrow{A^{''}} y, \Delta'$. (1)-(4) have been derived by a shorter derivation; thus we can replace the cut by cutting (1) and (3) on the one hand, and (4) and (2) on the other, which give respectively

$$(5)\ u : A^{'} \vdash u : A^{''} \text{ and } (6)\ u : A^{''} \vdash u : A^{'}.$$

Using (**EQ**) we obtain $\Gamma', x \xrightarrow{A^{'}} y \vdash \Delta', x \xrightarrow{A^{''}} y$.

(*d*) $F = x \xrightarrow{A} y$ is introduced on the left by (**ID**) rule, and it is introduced on the right by (**EQ**). Thus we have

$$\dfrac{\dfrac{u : A^{'} \vdash u : A \quad u : A \vdash u : A^{'}}{(2)\Gamma', x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y}\,(\mathbf{EQ}) \qquad \dfrac{(1)\Gamma', x \xrightarrow{A^{'}} y, x \xrightarrow{A} y, y : A \vdash \Delta}{x \xrightarrow{A} y, \Gamma', x \xrightarrow{A^{'}} y \vdash \Delta}\,(\mathbf{ID})}{\Gamma', x \xrightarrow{A^{'}} y \vdash \Delta}\,(\text{cut})$$

From (2) we can obtain a proof of no greater height of $(2^{'})$ $\Gamma', x \xrightarrow{A^{'}} y, y : A \vdash \Delta, x \xrightarrow{A} y$ by weakening (Theorem 4.10); moreover, by label substitution (Lemma 4.9) and weakening we can find a derivation of no greater height than $u : A^{'} \vdash u : A$'s of (3) $\Gamma', x \xrightarrow{A^{'}} y, y : A^{'} \vdash y : A, \Delta$. First, we replace the following cut by inductive hypothesis on the height:

$$\dfrac{(2^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, y : A \vdash \Delta, x \xrightarrow{A} y \qquad (1)\Gamma', x \xrightarrow{A^{'}} y, x \xrightarrow{A} y, y : A \vdash \Delta}{(4)\Gamma^{'}, x \xrightarrow{A^{'}} y, y : A \vdash \Delta}\,(cut)$$

From (4), we can find a proof of $(4^{'})$ $\Gamma^{'}, x \xrightarrow{A^{'}} y, y : A, y : A^{'} \vdash \Delta$ by weakening, thus we replace the initial cut as follows:

$$\dfrac{\dfrac{(3)\Gamma', x \xrightarrow{A^{'}} y, y : A^{'} \vdash y : A, \Delta \qquad (4^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, y : A, y : A^{'} \vdash \Delta}{\Gamma^{'}, x \xrightarrow{A^{'}} y, y : A^{'} \vdash \Delta}\,(cut)}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}\,(\mathbf{ID})$$

The above cut can be replaced by inductive hypothesis on the complexity of the cut formula;

(*e*) $F = x \xrightarrow{A} x$ is introduced on the left by (**EQ**) rule, and it is introduced on the right by (**MP**). Thus we have

$$\dfrac{\dfrac{(1)\Gamma \vdash \Delta^{'}, x \xrightarrow{A^{'}} x, x \xrightarrow{A} x, x : A}{\Gamma \vdash \Delta', x \xrightarrow{A'} x, x \xrightarrow{A} x}\,(\mathbf{MP}) \qquad \dfrac{u : A \vdash u : A' \quad u : A' \vdash u : A}{(2)\Gamma, x \xrightarrow{A} x \vdash \Delta', x \xrightarrow{A'} x}\,(\mathbf{EQ})}{\Gamma \vdash \Delta', x \xrightarrow{A'} x}\,(\text{cut})$$

From (2) we obtain a proof of no greater height of $(2')$ $\Gamma, x \xrightarrow{A} x \vdash \Delta', x \xrightarrow{A'} x, x : A$ by weakening, then we first replace the following cut by inductive hypothesis on the height:

$$\dfrac{(1)\Gamma \vdash \Delta', x \xrightarrow{A'} x, x \xrightarrow{A} x, x : A \qquad (2')\Gamma, x \xrightarrow{A} x \vdash \Delta', x \xrightarrow{A'} x, x : A}{(3)\Gamma \vdash \Delta', x \xrightarrow{A'} x, x : A}\,(cut)$$

from which we obtain a derivation of $(3')$ $\Gamma \vdash \Delta', x \xrightarrow{A'} x, x : A, x : A'$ by weakening. Moreover, by label substitution and weakening we can find a proof of (at most) the same height of $u : A \vdash u : A'$ of (4) $\Gamma, x : A \vdash \Delta', x \xrightarrow{A'} x, x : A'$. The initial cut can be replaced as follows (the cut below can be eliminated by inductive hypothesis on the complexity of the cut formula):

$$\begin{array}{c}\dfrac{(3')\Gamma \vdash \Delta', x \xrightarrow{A'} x, x : A, x : A' \qquad (4)\Gamma, x : A \vdash \Delta', x \xrightarrow{A'} x, x : A'}{\Gamma \vdash \Delta', x \xrightarrow{A'} x, x : A'}\,(cut) \\ \qquad\qquad (\mathbf{MP}) \\ \Gamma \vdash \Delta', x \xrightarrow{A'} x\end{array}$$

(f) $F = x \xrightarrow{A} x$ is introduced on the right by $(\mathbf{MP})$ rule and on the left by $(\mathbf{ID})$. Thus we have

$$\dfrac{\begin{array}{c}(1)\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A \\ (3)\Gamma \vdash \Delta, x \xrightarrow{A} x\end{array}(\mathbf{MP}) \qquad \begin{array}{c}(2)\Gamma, x \xrightarrow{A} x, x : A \vdash \Delta \\ (4)\Gamma, x \xrightarrow{A} x \vdash \Delta\end{array}(\mathbf{ID})}{\Gamma \vdash \Delta}\,(\text{cut})$$

By weakening, we can find derivations of $(3')$ $\Gamma, x : A \vdash \Delta, x \xrightarrow{A} x$ and $(4')$ $\Gamma, x \xrightarrow{A} x \vdash \Delta, x : A$ of no greater height than (3) and (4), respectively. Thus we replace the initial cut as follows:

$$\dfrac{\begin{array}{c}(1)\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A \\ (4')\Gamma, x \xrightarrow{A} x \vdash \Delta, x : A \\ \Gamma \vdash \Delta, x : A\end{array}(cut) \qquad \begin{array}{c}(3')\Gamma, x : A \vdash \Delta, x \xrightarrow{A} x \\ (2)\Gamma, x \xrightarrow{A} x, x : A \vdash \Delta \\ \Gamma, x : A \vdash \Delta\end{array}(cut)}{\Gamma \vdash \Delta}\,(cut)$$

The lower cut can be replaced by inductive hypothesis on the complexity of the cut formula, the other ones by inductive hypothesis on the height.

(g) $F = x \xrightarrow{A} y$ is derived on the left by $(\mathbf{CS})$ and on the right by $(\mathbf{EQ})$. Thus we have (we denote with $\Sigma(u)$ the substitution $\Sigma[x/u, y/u]$):

$$\dfrac{\dfrac{(1)u : A \vdash u : A' \quad (2)u : A' \vdash u : A}{(5)\Gamma', x \xrightarrow{A'} y \vdash \Delta, x \xrightarrow{A} y}(\mathbf{EQ}) \qquad \dfrac{\begin{array}{c}(3)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta, x : A \\ (4)\Gamma'(u), u \xrightarrow{A'} u, u \xrightarrow{A} u \vdash \Delta(u)\end{array}}{\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta}(\mathbf{CS})}{\Gamma', x \xrightarrow{A'} y \vdash \Delta}(cut)$$

First, we can replace the cut below:

$$\dfrac{(5')\Gamma', x \xrightarrow{A'} y \vdash \Delta, x \xrightarrow{A} y, x : A \qquad (3)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta, x : A}{(6)\Gamma', x \xrightarrow{A'} y \vdash \Delta, x : A}(cut)$$

by inductive hypothesis on the height. $(5')$ is obtained from (5) since weakening is height-preserving admissible. We replace the initial cut as follows:

$$\dfrac{\dfrac{\begin{array}{c}(6')\Gamma', x \xrightarrow{A'} y \vdash \Delta, x : A, x : A' \\ (1')\Gamma', x \xrightarrow{A'} y, x : A \vdash \Delta, x : A'\end{array}}{\Gamma', x \xrightarrow{A'} y \vdash \Delta, x : A'}(cut) \qquad \dfrac{\begin{array}{c}(5'')\Gamma'(u), u \xrightarrow{A'} u \vdash \Delta(u), u \xrightarrow{A} u \\ (4)\Gamma'(u), u \xrightarrow{A'} u, u \xrightarrow{A} u \vdash \Delta(u)\end{array}}{\Gamma'(u), u \xrightarrow{A'} u \vdash \Delta(u)}(cut)}{\Gamma', x \xrightarrow{A'} y \vdash \Delta}(\mathbf{CS})$$

where $(6')$ is obtained from (6) by weakening, $(5'')$ from (5) by label substitution and $(1')$ from (1) by weakening and label substitution. The cut on the left can be replaced by inductive hypothesis on the complexity of the cut formula; the cut on the right can be replaced by inductive hypothesis on the height.

In the systems containing (**CEM**) the standard proof does not work in one case: when a transition $x \xrightarrow{A} y$ is the cut formula and it is introduced by (**EQ**) in one premise of cut and by (**CEM**) in the other, one needs to apply cut *two times on the same formula* $x \xrightarrow{A} y$ to replace the initial cut.

Therefore, in order to prove the admissibility of cut for systems with (**CEM**), we proceed as follows. First of all, we need to extend our notation by representing with $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ a sequent containing *any* number of transitions labelled with the formula A; moreover, given $u : A \vdash u : A'$ and $u : A' \vdash u : A$, we denote with $\Gamma^\star \vdash \Delta^\star$ the sequent obtained by replacing *any* number of transitions labelled with A with the same transitions labelled with A' in $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$.

Definition 4.14. *We denote with*

$$\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$$

a sequent $\Gamma \vdash \Delta$ *such that* n *transitions* $x_1 \xrightarrow{A} y_1, x_2 \xrightarrow{A} y_2, ..., x_n \xrightarrow{A} y_n \in \Gamma$ *and* m *transitions* $u_1 \xrightarrow{A} v_1, u_2 \xrightarrow{A} v_2, ..., u_n \xrightarrow{A} v_m \in \Delta$*, where* $n, m \geq 0$*.*

Definition 4.15. *If* $u : A \vdash u : A'$ *and* $u : A' \vdash u : A$ *are derivable, then we denote* *with*

$$\Gamma^\star \vdash \Delta^\star$$

the sequent obtained by replacing in $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ any *number of transitions* $x_i \xrightarrow{A} y_i$ *with* $x_i \xrightarrow{A'} y_i$ *and* any *number of transitions* $u_j \xrightarrow{A} u_j$ *with* $u_j \xrightarrow{A'} u_j$*.*

As an example, given $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j] = x : A, x \xrightarrow{A} y, y \xrightarrow{A} z \vdash$, we use $\Gamma^\star \vdash \Delta^\star$ to denote one of the following sequents: (1) $x : A, x \xrightarrow{A} y, y \xrightarrow{A} z \vdash$, (2) $x : A, x \xrightarrow{A'} y, y \xrightarrow{A} z \vdash$, (3) $x : A, x \xrightarrow{A} y, y \xrightarrow{A'} z \vdash$, (4) $x : A, x \xrightarrow{A'} y, y \xrightarrow{A'} z \vdash$. In (1) no transitions $x_i \xrightarrow{A} y_i$ have been replaced, in (2) the transition $x \xrightarrow{A} y$ has been replaced by $x \xrightarrow{A'} y$, and so on.

We prove that cut is admissible in systems SeqCEM{+CS}{+ID} by "splitting" the notion of cut in two propositions:

(A) if $\Gamma \vdash \Delta, F$ and $\Gamma, F \vdash \Delta$ are derivable, so $\Gamma \vdash \Delta$ (cut);

(B) if (I) $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$, (II) $u : A \vdash u : A'$ and (III) $u : A' \vdash u : A$ are derivable, so $\Gamma^\star \vdash \Delta^\star$

whose proof is by double mutual induction on the complexity of the cut formula and on the height of the derivation. To prove (A), the induction on the height is intended as usual as the sum of the height of the premises of the cut inference; to prove (B), the induction on the height is intended as the height of the derivation of $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$. We have several cases:

- (Base for (A)): if one of the two premises is an axiom, then either $\Gamma \vdash \Delta$ is an axiom, or the premise which is not an axiom contains two copies of F and $\Gamma \vdash \Delta$ can be obtained by contraction, which is admissible (Theorem 4.12);
- (Base for (B)): if $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ is an axiom, then we distinguish two subcases:
  - ⋆ $x : P \in \Gamma[x_i \xrightarrow{A} y_i]$ and $x : P \in \Delta[u_j \xrightarrow{A} v_j]$: we conclude that $\Gamma^\star \vdash \Delta^\star$ is derivable, since $x : P \in \Gamma^\star$ and $x : P \in \Delta^\star$ (only transition formulas can be modified by applying the proposition (B));
  - ⋆ $x : \bot \in \Gamma[x_i \xrightarrow{A} y_i]$: as in the other case, we conclude since $x : \bot \in \Gamma^\star$;
- (Inductive step for (A)): we distinguish all cases:
  1. the last step of *one* of the two premises is obtained by a rule in which F is *not* the principal formula. We distinguish two cases.
     (i) The sequent where F is not principal is derived by any rule (**R**), except the (**EQ**) rule. This case is standard, we can permute (**R**) over the cut: i.e. we cut the premise(s) of (**R**) and then we apply (**R**) to the result of cut. As an example, consider the case when $F = x \xrightarrow{A} y$ is the cut formula, and it is principal only in the right derivation, introduced (forward) by an application of (**CEM**). In the left derivation (**CS**) is applied to another transition $x \xrightarrow{A'} y$ (consider $\Sigma(u) = \Sigma[x/u, y/u]$):

$$\dfrac{\dfrac{\begin{array}{c}(1)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y, x : A^{'} \\ (2)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u), u \xrightarrow{A} u\end{array}}{(5)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y}\,(\mathbf{CS}) \qquad \dfrac{\begin{array}{c}(3)\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z \\ (4)(\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta)[y/v, z/v]\end{array}}{(6)\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CEM})}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}\,(cut)$$

From (6) we obtain a proof of (at most) the same height of $(6^{'})$ $\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x : A^{'}$, by Theorem 4.10. Applying Lemma 4.9 to (6) we obtain a proof of no greater height of $(6^{''})$ $\Gamma^{'}(u), u \xrightarrow{A^{'}} u, u \xrightarrow{A} u \vdash \Delta(u)$. For the inductive hypothesis on the height, we cut (1) with $(6^{'})$, obtaining (7) $\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A^{'}$; then we cut (2) with $(6^{''})$, obtaining a proof of (8) $\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u)$. The initial cut is replaced as follows:

$$\dfrac{\dfrac{\begin{array}{c}(1)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y, x : A^{'} \\ (6^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x : A^{'}\end{array}}{(7)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A^{'}}\,(cut) \qquad (8)\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u)}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}\,(\mathbf{CS})$$

(ii) If one of the sequents, say $\Gamma \vdash \Delta, F$ is obtained by the (**EQ**) rule, where F is not principal, then also $\Gamma \vdash \Delta$ is derivable by the (**EQ**) rule and we are done;

2. F is the principal formula in the last step of *both* derivations of the premises of the cut inference. There are six subcases: F is introduced (a) by (→ **L**), (→ **R**), (b) by (⇒ **L**), (⇒ **R**), (c) by (**EQ**), (d) by (**ID**) on the left and by (**EQ**) on the right, (e) by (**CS**) on the left and by (**EQ**) on the right, and (f) by (**CEM**) on the left and by (**EQ**) on the right. The list is exhaustive, since we do not consider systems allowing both (**MP**) and (**CEM**).

(a) $F = x : A \rightarrow B$ is introduced by (→ **L**) on the left and by (→ **R**) on the right as follows:

$$\dfrac{\dfrac{(1)\Gamma, x : A \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : A \rightarrow B}\,(\rightarrow \mathbf{R}) \qquad \dfrac{(2)\Gamma \vdash \Delta, x : A \qquad (3)\Gamma, x : B \vdash \Delta}{\Gamma, x : A \rightarrow B \vdash \Delta}\,(\rightarrow \mathbf{L})}{\Gamma \vdash \Delta}\,(cut)$$

This cut is replaced by two cuts on the subformulas A and B as follows (the sequent $(2^{'})$ is obtained by weakening from (2)):

$$\dfrac{\dfrac{(2^{'})\Gamma \vdash \Delta, x : A, x : B \qquad (1)\Gamma, x : A \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : B}\,(cut) \qquad (3)\Gamma, x : B \vdash \Delta}{\Gamma \vdash \Delta}\,(cut)$$

(b) $F = x : A \Rightarrow B$ is introduced by ($\Rightarrow$ **R**) and ($\Rightarrow$ **L**). Then we have

$$\dfrac{\genfrac{}{}{0pt}{}{(1)\Gamma, x \xrightarrow{A} z \vdash \Delta, z : B}{(2)\Gamma \vdash \Delta, x : A \Rightarrow B}\;(\Rightarrow \mathbf{R}) \qquad \dfrac{(3)\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \quad (4)\Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{(5)\Gamma, x : A \Rightarrow B \vdash \Delta}\;(\Rightarrow \mathbf{L})}{\Gamma \vdash \Delta}\;(cut)$$

where z does not occur in Γ, Δ and $z \neq x$; By Lemma 4.9, we obtain that $(1^{'})$ $\Gamma, x \xrightarrow{A} y \vdash y : B, \Delta$ is derivable by a derivation of no greater height than (1); moreover, we can apply the height-preserving admissibility of weakening (Theorem 4.10) to (2) in order to obtain a proof of no greater height of $(2^{'})$ $\Gamma \vdash \Delta, x : A \Rightarrow B, x \xrightarrow{A} y$ and of $(2^{''})$ $\Gamma, y : B \vdash \Delta, x : A \Rightarrow B$.

First, we can make the following cut, which uses the inductive hypothesis on the height:

$$\dfrac{(2^{'})\Gamma \vdash \Delta, x : A \Rightarrow B, x \xrightarrow{A} y \qquad (3)\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y}{(6)\Gamma \vdash \Delta, x \xrightarrow{A} y}\;(cut)$$

Applying the height-preserving admissibility of weakening to (6), we have a proof of no greater height of $(6^{'})$ $\Gamma \vdash \Delta, x \xrightarrow{A} y, y : B$. Thus we can replace the initial cut as follows:

$$\dfrac{\genfrac{}{}{0pt}{}{\genfrac{}{}{0pt}{}{(2^{''})\Gamma, y : B \vdash \Delta, x : A \Rightarrow B}{(4)\Gamma, x : A \Rightarrow B, y : B \vdash \Delta}}{\Gamma, y : B \vdash \Delta}\;(cut) \qquad \genfrac{}{}{0pt}{}{\genfrac{}{}{0pt}{}{(1^{'})\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{(6^{'})\Gamma \vdash \Delta, x \xrightarrow{A} y, y : B}}{\Gamma \vdash \Delta, y : B}\;(cut)}{\Gamma \vdash \Delta}\;(cut)$$

The upper cut on the left uses the induction hypothesis on the height, the others the induction hypothesis on the complexity of the cut formula.

(c) $F = x \xrightarrow{A} y$ is introduced by (**EQ**) in both premises, we have

$$\dfrac{\genfrac{}{}{0pt}{}{(1)\, u : A^{'} \vdash u : A \quad (2)\, u : A \vdash u : A^{'}}{\Gamma', x \xrightarrow{A^{'}} y \vdash x \xrightarrow{A} y, \Delta}\;(\mathbf{EQ}) \qquad \genfrac{}{}{0pt}{}{(3)\, u : A \vdash u : A^{''} \quad (4)\, u : A^{''} \vdash u : A}{\Gamma, x \xrightarrow{A} y \vdash x \xrightarrow{A^{''}} y, \Delta'}\;(\mathbf{EQ})}{\Gamma', x \xrightarrow{A^{'}} y \vdash x \xrightarrow{A^{''}} y, \Delta'}\;(cut)$$

where $\Gamma = \Gamma', x \xrightarrow{A^{'}} y$, $\Delta = x \xrightarrow{A^{''}} y, \Delta'$. (1)-(4) have been derived by a shorter derivation; thus we can replace the cut by cutting (1) and (3) on the one hand, and (4) and (2) on the other, which give respectively

$$(5)\ u : A^{'} \vdash u : A^{''} \text{ and } (6)\ u : A^{''} \vdash u : A^{'}.$$

Using (**EQ**) we obtain $\Gamma', x \xrightarrow{A^{'}} y \vdash \Delta', x \xrightarrow{A^{''}} y$.

(d) $F = x \xrightarrow{A} y$ is introduced on the left by (**ID**) rule, and it is introduced on the right by (**EQ**). Thus we have

$$\dfrac{\dfrac{u:A' \vdash u:A \quad u:A \vdash u:A'}{(2)\Gamma', x \xrightarrow{A'} y \vdash \Delta, x \xrightarrow{A} y}(\mathbf{EQ}) \qquad \dfrac{(1)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y, y:A \vdash \Delta}{x \xrightarrow{A} y, \Gamma', x \xrightarrow{A'} y \vdash \Delta}(\mathbf{ID})}{\Gamma', x \xrightarrow{A'} y \vdash \Delta}(\text{cut})$$

From (2) we can obtain a proof of no greater height of $(2')\Gamma', x \xrightarrow{A'} y, y:A \vdash \Delta, x \xrightarrow{A} y$ by weakening (Theorem 4.10); moreover, by label substitution (Lemma 4.9) and weakening we can find a derivation of no greater height than $u:A' \vdash u:A$'s of $(3)\Gamma', x \xrightarrow{A'} y, y:A' \vdash y:A, \Delta$. First, we replace the following cut by inductive hypothesis on the height:

$$\dfrac{(2')\Gamma', x \xrightarrow{A'} y, y:A \vdash \Delta, x \xrightarrow{A} y \qquad (1)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y, y:A \vdash \Delta}{(4)\Gamma', x \xrightarrow{A'} y, y:A \vdash \Delta}(cut)$$

From (4), we can find a proof of $(4')\Gamma', x \xrightarrow{A'} y, y:A, y:A' \vdash \Delta$ by weakening, thus we replace the initial cut as follows:

$$\dfrac{\dfrac{(3)\Gamma', x \xrightarrow{A'} y, y:A' \vdash y:A, \Delta \qquad (4')\Gamma', x \xrightarrow{A'} y, y:A, y:A' \vdash \Delta}{\Gamma', x \xrightarrow{A'} y, y:A' \vdash \Delta}(cut)}{\Gamma', x \xrightarrow{A'} y \vdash \Delta}(\mathbf{ID})$$

The above cut can be replaced by inductive hypothesis on the complexity of the cut formula;

(*e*) $F = x \xrightarrow{A} y$ is derived on the left by (**CS**) and on the right by (**EQ**). Thus we have (we denote with $\Sigma(u)$ the substitution $\Sigma[x/u, y/u]$):

$$\dfrac{\dfrac{(1)u:A \vdash u:A' \qquad (2)u:A' \vdash u:A}{(5)\Gamma', x \xrightarrow{A'} y \vdash \Delta, x \xrightarrow{A} y}(\mathbf{EQ}) \qquad \dfrac{\begin{matrix}(3)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta, x:A \\ (4)\Gamma'(u), u \xrightarrow{A'} u, u \xrightarrow{A} u \vdash \Delta(u)\end{matrix}}{\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta}(\mathbf{CS})}{\Gamma', x \xrightarrow{A'} y \vdash \Delta}(cut)$$

First, we can replace the cut below:

$$\dfrac{(5')\Gamma', x \xrightarrow{A'} y \vdash \Delta, x \xrightarrow{A} y, x:A \qquad (3)\Gamma', x \xrightarrow{A'} y, x \xrightarrow{A} y \vdash \Delta, x:A}{(6)\Gamma', x \xrightarrow{A'} y \vdash \Delta, x:A}(cut)$$

by inductive hypothesis on the height. $(5')$ is obtained from (5) since weakening is height-preserving admissible. We replace the initial cut as follows:

$$\dfrac{\dfrac{(6^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A, x : A^{'} \qquad (1^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, x : A \vdash \Delta, x : A^{'}}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x : A^{'}}(cut) \qquad \dfrac{(5^{''})\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u), u \xrightarrow{A} u \qquad (4)\Gamma^{'}(u), u \xrightarrow{A^{'}} u, u \xrightarrow{A} u \vdash \Delta(u)}{\Gamma^{'}(u), u \xrightarrow{A^{'}} u \vdash \Delta(u)}(cut)}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}(\mathbf{CS})$$

where $(6^{'})$ is obtained from (6) by weakening, $(5^{''})$ from (5) by label substitution and $(1^{'})$ from (1) by weakening and label substitution. The cut on the left can be replaced by inductive hypothesis on the complexity of the cut formula; the cut on the right can be replaced by inductive hypothesis on the height.

(f) $F = x \xrightarrow{A} y$ is derived on the left by (**CEM**) and on the right by (**EQ**). Thus we have:

$$\dfrac{\dfrac{(1)u : A \vdash u : A^{'} \qquad (2)u : A^{'} \vdash u : A}{(5)\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y}(\mathbf{EQ}) \qquad \dfrac{(3)\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z \qquad (4)(\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta)[y, z/u]}{\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta}(\mathbf{CEM})}{\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta}(cut)$$

where $y \neq z$. Since we have (1) and (2), by mutual induction we can apply proposition (B) on (3), obtaining a derivation of the sequent $(3^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A^{'}} z$ (we replace $x \xrightarrow{A} y$ with $x \xrightarrow{A^{'}} y$ and $x \xrightarrow{A} z$ with $x \xrightarrow{A^{'}} z$). $(5^{'})(\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y)[y, z/u]$ is obtained by Lemma 4.9 and has no greater height than (5). We can replace the above cut as follows:

$$\dfrac{(3^{'})\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A^{'}} z \qquad \dfrac{(5^{'})(\Gamma^{'}, x \xrightarrow{A^{'}} y \vdash \Delta, x \xrightarrow{A} y)[y, z/u] \qquad (4)(\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A} y \vdash \Delta)[y, z/u]}{(6)(\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A^{'}} y \vdash \Delta)[y, z/u]}(cut)}{\Gamma^{'}, x \xrightarrow{A^{'}} y, x \xrightarrow{A^{'}} y \vdash \Delta}(\mathbf{CEM})$$

from which we conclude by contraction, which is admissible (Theorem 4.12). Notice that one instance of $x \xrightarrow{A^{'}} y$ is introduced in (6) by weakening (Theorem 4.10).

- (Inductive step for (B)): we consider all cases:

1. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of ($\rightarrow$ **R**), i.e.:

$$\dfrac{\Gamma[x_i \xrightarrow{A} y_i], x : A \vdash \Delta[u_j \xrightarrow{A} v_j], x : B}{\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j], x : A \rightarrow B}(\rightarrow \mathbf{R})$$

By inductive hypothesis, we have that $\Gamma^\star, x : A \vdash \Delta^\star, x : B$ is derivable, from which we conclude by an application of $(\rightarrow \mathbf{R})$;

2. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of $(\rightarrow \mathbf{L})$, i.e.:

$$\frac{\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j], x : A \qquad \Gamma[x_i \xrightarrow{A} y_i], x : B \vdash \Delta[u_j \xrightarrow{A} v_j]}{\Gamma[x_i \xrightarrow{A} y_i], x : A \rightarrow B \vdash \Delta[u_j \xrightarrow{A} v_j]}\ (\rightarrow \mathbf{L})$$

We can apply the inductive hypothesis on both premises, obtaining derivations for $\Gamma^\star \vdash \Delta^\star, x : A$ and $\Gamma^\star, x : B \vdash \Delta^\star$, from which we obtain a derivation of $\Gamma^\star, x : A \rightarrow B \vdash \Delta^\star$ by an application of $(\rightarrow \mathbf{L})$;

3. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of $(\Rightarrow \mathbf{R})$; the proof is ended as follows:

$$\frac{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j], y : B}{\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j], x : A \Rightarrow B}\ (\Rightarrow \mathbf{R})$$

By inductive hypothesis on the premise, we can replace any transition $x_i \xrightarrow{A} y_i$ with $x \xrightarrow{A^{'}} y$ and *we do not replace the transition* $x \xrightarrow{A} y$: we can conclude as follows:

$$\begin{array}{c} \Gamma^\star, x \xrightarrow{A} y \vdash \Delta^\star, y : B \\ \Gamma^\star \vdash \Delta^\star, x : A \Rightarrow B \end{array}\ (\Rightarrow \mathbf{R})$$

4. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of $(\Rightarrow \mathbf{L})$; the proof is ended as follows:

$$\frac{\Gamma[x_i \xrightarrow{A} y_i], x : A \Rightarrow B \vdash \Delta[u_j \xrightarrow{A} v_j], x \xrightarrow{A} y \qquad \Gamma[x_i \xrightarrow{A} y_i], x : A \Rightarrow B, y : B \vdash \Delta[u_j \xrightarrow{A} v_j]}{\Gamma[x_i \xrightarrow{A} y_i], x : A \Rightarrow B \vdash \Delta[u_j \xrightarrow{A} v_j]}\ (\Rightarrow \mathbf{L})$$

As we made in the previous case, we apply the inductive hypothesis on the two premises, obtaining proofs of the sequents $\Gamma^\star, x : A \Rightarrow B \vdash \Delta^\star, x \xrightarrow{A} y$ and $\Gamma, x : A \Rightarrow B, y : B \vdash \Delta^\star$, from which we conclude by an application of $(\Rightarrow \mathbf{L})$;

5. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of $(\mathbf{EQ})$, applied on transitions $x \xrightarrow{F} y \in \Gamma$ and $x \xrightarrow{F^{'}} y \in \Delta$. If $F, F^{'}$ are both different from A, then we have that $(1) x \xrightarrow{F} y \vdash x \xrightarrow{F^{'}} y$ is derivable, thus we conclude that $\Gamma^\star \vdash \Delta^\star$, since we can add any $x_i \xrightarrow{A^{'}} y_i$ and $u_j \xrightarrow{A^{'}} v_j$ to (1) by weakening, which is admissible. If $F = A$ and $F^{'} = A^{'}$, then we can obviously conclude, since both $\Gamma^*, x \xrightarrow{A^{'}} y \vdash \Delta^*, x \xrightarrow{A} y$ and $\Gamma^*, x \xrightarrow{A^{'}} y \vdash \Delta^*, x \xrightarrow{A^{'}} y$ are derivable. Otherwise, we have that $F = A$ and $F^{'} \neq A^{'}$ (the case when $F^{'} = A$ and $F \neq A^{'}$ is symmetric), thus:

$$\frac{(2) u : A \vdash u : A^{''} \qquad (3) u : A^{''} \vdash u : A}{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j], x \xrightarrow{A^{''}} y}\ (\mathbf{EQ})$$

We have also that $(*)u : A \vdash u : A^{'}$ and $(**)u : A^{'} \vdash u : A$. We can apply the proposition (A) as follows (inductive hypothesis on the complexity of the cut formula):

$$\dfrac{(**)u : A^{'} \vdash u : A \qquad (2)u : A \vdash u : A^{''}}{(4)u : A^{'} \vdash u : A^{''}}\ (cut)$$

$$\dfrac{(3)u : A^{''} \vdash u : A \qquad (*)u : A \vdash u : A^{'}}{(5)u : A^{''} \vdash u : A^{'}}\ (cut)$$

From (4) and (5) we obtain a proof of $x \xrightarrow{A^{'}} y \vdash x \xrightarrow{A^{''}} y$ and, by the admissibility of weakening, we obtain a proof of $\Gamma^{\star}, x \xrightarrow{A^{'}} y \vdash \Delta^{\star}, x \xrightarrow{A^{''}} y$, and we are done. If we want to derive $\Gamma^{*}, x \xrightarrow{A} y \vdash \Delta^{*}, x \xrightarrow{A^{''}} y$ we have the following proof: $x \xrightarrow{A} y \vdash x \xrightarrow{A^{''}} y$ derives (forward) from (2) and (3) by (**EQ**), and $\Gamma^{*}, x \xrightarrow{A} y \vdash \Delta^{*}, x \xrightarrow{A^{''}} y$ is obtained by weakening;

6. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of (**ID**): if (**ID**) has $x \xrightarrow{F} y$, with F different from A, as a principal formula, then we can obviously conclude by applying the inductive hypothesis on the premise. The only interesting case is when (**ID**) is applied to $x \xrightarrow{A} y$ as follows:

$$\dfrac{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y, y : A \vdash \Delta[u_j \xrightarrow{A} v_j]}{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j]}\ (\mathbf{ID})$$

We can apply the inductive hypothesis on the premise, obtaining a proof of $(1)\Gamma^{\star}, x \xrightarrow{A^{'}} y, y : A \vdash \Delta^{\star}$, to which we can apply the proposition (A) (inductive hypothesis again on the complexity of the cut formula, we omit necessary weakenings and label substitutions) and conclude by an application of (**ID**) as follows:

$$\dfrac{\dfrac{(**)y : A^{'} \vdash y : A \qquad (1)\Gamma^{\star}, x \xrightarrow{A^{'}} y, y : A \vdash \Delta^{\star}}{\Gamma^{\star}, x \xrightarrow{A^{'}} y, y : A^{'} \vdash \Delta^{\star}}\ (cut)}{\Gamma^{\star}, x \xrightarrow{A^{'}} y \vdash \Delta^{\star}}\ (\mathbf{ID})$$

7. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of (**CS**): as for (**ID**), the only interesting case is when (**CS**) has a transition $x \xrightarrow{A} y$ as a principal formula. The proof is ended as follows:

$$\dfrac{(1)\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j], x : A \qquad (2)(\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j])[x, y/u]}{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j]}\ (\mathbf{CS})$$

We can apply the inductive hypothesis on both premises, obtaining a proof of $(1')\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star, x : A$ and $(2')(\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star)[x, y/u]$, respectively. We conclude by applying the proposition (A) (inductive hypothesis on the complexity of the cut formula) and by an application of (**CS**) as follows:

$$\dfrac{\dfrac{(1')\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star, x : A \qquad (*)x : A \vdash x : A'}{\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star, x : A'}\,(cut) \qquad (2')(\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star)[x, y/u]}{\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star}\,(\mathbf{CS})$$

8. $\Gamma[x_i \xrightarrow{A} y_i] \vdash \Delta[u_j \xrightarrow{A} v_j]$ derives from an application of (**CEM**) as follows:

$$\dfrac{(1)\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j], x \xrightarrow{A} z \qquad (2)(\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j])[y, z/u]}{\Gamma[x_i \xrightarrow{A} y_i], x \xrightarrow{A} y \vdash \Delta[u_j \xrightarrow{A} v_j]}\,(\mathbf{CEM})$$

We can apply the inductive hypothesis on the height to both premises and we can easily conclude as follows:

$$\dfrac{(1')\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star, x \xrightarrow{A'} z \qquad (2')(\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star)[y, z/u]}{\Gamma^\star, x \xrightarrow{A'} y \vdash \Delta^\star}\,(\mathbf{CEM})$$

■

As mentioned above, we are not able to prove that in systems allowing both (**CEM**) and (**MP**), i.e. SeqCEM{+CS}{+ID}+MP, cut is admissible. Indeed, extending the proof of cut elimination presented above, we have to consider the case when the cut formula is introduced by (**CEM**) on the left and by (**MP**) on the right as follows:

$$\dfrac{\dfrac{\Gamma \vdash \Delta, x : A}{\Gamma \vdash \Delta, x \xrightarrow{A} x}\,(\mathbf{MP}) \qquad \dfrac{\Gamma, x \xrightarrow{A} x \vdash \Delta, x \xrightarrow{A} z \qquad (\Gamma, x \xrightarrow{A} x \vdash \Delta)[x/u, z/u]}{\Gamma, x \xrightarrow{A} x \vdash \Delta}\,(\mathbf{CEM})}{\Gamma \vdash \Delta}\,(cut)$$

At the moment, we are not able to conclude the proof in this situation, nor to give a counterexample showing that an explicit cut rule is needed to make the calculus complete. We strongly conjecture that cut is admissible even in these systems, and we intend to prove it in our future research.

4.3.2 Soundness and Completeness of SeqS

SeqS calculi are sound and complete with respect to the semantics.

Theorem 4.16 (Soundness)**.** *If* $\Gamma \vdash \Delta$ *is derivable in SeqS then it is valid in the corresponding system.*

Proof. By induction on the height of a derivation of $\Gamma \vdash \Delta$. As an example, we examine the cases of ($\Rightarrow$ **R**), (**MP**), (**CS**) and (**CEM**). The other cases are left to the reader.

- ($\Rightarrow$ **R**) Let $\Gamma \vdash \Delta, x : A \Rightarrow B$ be derived from (1) $\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B$, where y does not occur in Γ, Δ and it is different from x. By induction hypothesis we know that the latter sequent is valid. Suppose the former is not, and that it is not valid in a model $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$, via a mapping I, so that we have:

  $$\mathcal{M} \models_I F \text{ for every } F \in \Gamma,\ \mathcal{M} \not\models_I F \text{ for any } F \in \Delta \text{ and } M \not\models_I x : A \Rightarrow B.$$

  As $\mathcal{M} \not\models_I x : A \Rightarrow B$ there exists $w \in f(I(x), [A]) - [B]$. We can define an interpretation $I'(z) = I(z)$ for $z \neq y$ and $I'(y) = w$. Since y does not occur in Γ, Δ and is different from x, we have that $\mathcal{M} \models_{I'} F$ for every $F \in \Gamma$, $\mathcal{M} \not\models_{I'} F$ for any $F \in \Delta$, $\mathcal{M} \not\models_{I'} y : B$ and $\mathcal{M} \models_{I'} x \xrightarrow{A} y$, against the validity of (1).

- (**MP**) Let $\Gamma \vdash \Delta, x \xrightarrow{A} x$ be derived from (2) $\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A$. Let (2) be valid and let $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ be a model satisfying the MP condition. Suppose that for one mapping I, $\mathcal{M} \models_I F$ for every $F \in \Gamma$, then by the validity of (2) either $\mathcal{M} \models_I G$ for some $G \in \Delta$, or $\mathcal{M} \models_I x \xrightarrow{A} x$, or $\mathcal{M} \models_I x : A$. In the latter case, we have $I(x) \in [A]$, thus $I(x) \in f(I(x), [A])$, by MP, this means that $\mathcal{M} \models_I x \xrightarrow{A} x$.

- (**CS**) Let (3)$\Gamma, x \xrightarrow{A} y \vdash \Delta$, with $x \neq y$, be derived from (4)$\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A$ and (5)$\Gamma[x/u, y/u], u \xrightarrow{A} u \vdash \Delta[x/u, y/u]$, where u does not occur in Γ, Δ. Suppose that (4) and (5) are valid, whereas (3) is not, considering a model $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ satisfying the CS condition. Therefore, there is a mapping I such that $\mathcal{M} \models_I F$ for every $F \in \Gamma$, $\mathcal{M} \models_I x \xrightarrow{A} y$ (i.e. $I(y) \in f(I(x), [A])$) and $\mathcal{M} \not\models_I G$ for every $G \in \Delta$. We distinguish two cases:

  - ⋆ $I(x) \notin [A]$: in this case, we have that $\mathcal{M} \not\models_I x : A$, against the validity of (4);
  - ⋆ $I(x) \in [A]$: we observe that $f(I(x), [A]) \subseteq \{I(x)\}$, since $\mathcal{M}$ respects the CS condition. We have also that $I(y) \in f(I(x), [A])$, then it can be only $I(x) = I(y)$: say $w = I(x) = I(y)$. We introduce another mapping $I^{'}$ as follows: $I^{'}(u) = w$, $I^{'}(v) = I(v)$ for every label different from u. Obviously, $\mathcal{M} \models_{I'} F$ for every $F \in \Gamma[x/u, y/u]$, and $\mathcal{M} \not\models_{I'} G$ for every $G \in \Delta[x/u, y/u]$, but $\mathcal{M} \models_{I'} u \xrightarrow{A} u$, since $I^{'}(u) = w \in f(w, [A])$, against the validity of (5).

- (**CEM**) Let (8)$\Gamma, x \xrightarrow{A} y \vdash \Delta$ be derived from (6)$\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z$ and (7)$(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]$, with $z \neq y$. Suppose (6) and (7) are valid, whereas (8) is not. $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ respects the CEM condition. Then, one can find a mapping I such that $\mathcal{M} \models_I F$ for every $F \in \Gamma$, $\mathcal{M} \not\models_I G$ for every $G \in \Delta$ and $\mathcal{M} \models_I x \xrightarrow{A} y$, thus $I(y) \in f(I(x), [A])$. We distinguish two cases:

  - ⋆ $I(y) \neq I(z)$: since $\mathcal{M}$ respects CEM, we have that $\mid f(I(x), [A]) \mid \leq 1$. In this case, $f(I(x), [A]) = \{I(y)\}$, then $I(z) \notin f(I(x), [A])$. We can conclude $\mathcal{M} \not\models_I x \xrightarrow{A} z$, against the validity of (6);
  - ⋆ $I(z) = I(y) = w$, therefore $f(I(x), [A]) = \{w\}$. We introduce another mapping $I^{'}$ in this way: $I^{'}(u) = w$; $I^{'}(v) = I(v)$ for every label v different from u. Obviously, $\mathcal{M} \models_{I'} F$ for every $F \in \Gamma[y/u, z/u]$, and $\mathcal{M} \not\models_{I'} G$ for every $G \in \Delta[y/u, z/u]$. However, $\mathcal{M} \models_{I'} x \xrightarrow{A} y[y/u, z/u]$ since $I^{'}(u) = w \in f(I^{'}(x), [A])$, against the validity of (7).

- ($\Rightarrow$ **L**) Let (9) $\Gamma, x : A \Rightarrow B \vdash \Delta$ be derived from (10) $\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y$ and (11) $\Gamma, x : A \Rightarrow B, y : B \vdash \Delta$. By inductive hypothesis, we have that both (10) and (11) are valid. By absurd, suppose that (9) is not valid, that is to say there are a model $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ and a mapping I such that $\mathcal{M} \models_I F$ for every $F \in \Gamma$, as well as $\mathcal{M} \models_I x : A \Rightarrow B$, i.e. $f(I(x), [A]) \subseteq [B]$, and $\mathcal{M} \not\models_I G$ for every $G \in \Delta$. Since (10) is valid, then it is also valid in $\mathcal{M}$, therefore we have that $\mathcal{M} \models_I x \xrightarrow{A} y$ (otherwise (10) is not valid in $\mathcal{M}$, since $\mathcal{M}$ satisfies all the formulas in the left-hand side via I and falsifies all the formulas in the right-hand side). Since $\mathcal{M} \models_I x \xrightarrow{A} y$, i.e. $I(y) \in f(I(x), [A])$, and $f(I(x), [A]) \subseteq [B]$, we can conclude that $I(y) \in [B]$, that is to say $\mathcal{M} \models_I y : B$. Therefore, we have that $\mathcal{M} \models_I F$ for every $F \in \Gamma$, $\mathcal{M} \models_I x : A \Rightarrow B$, $\mathcal{M} \models_I y : B$, and $\mathcal{M} \not\models_I G$ for every $G \in \Delta$: this means that (11) $\Gamma, x : A \Rightarrow B, y : B \vdash \Delta$ is not valid in $\mathcal{M}$, against the validity of (11).

■

Let us now prove the completeness of the SeqS calculi. As usual, completeness is an easy consequence of the admissibility of cut. It is worth noticing that one can give a semantic proof of completeness, however as a difference with modal logics, the proof is considerably more complex and require nonetheless the cut rule (see [OS00] for a semantic completeness proof of a tableau calculus for CK). We explain intuitively the difficulty. The usual way to prove completeness semantically is by contraposition, that is to say to extract a counter model from a failed branch of a (suitable) proof tree. To this purpose one needs to "saturate" a branch by applying the rules as much as possible. However the model being constructed must satisfy the normality condition, i.e. if $[A] = [A']$ then it must be $f(A, x) = f(A', x)$, or equivalently, the selection function must be well-defined on arbitrary subsets of worlds; to ensure this property, a simple branch saturation is not enough. One has to consider in the saturation process other formulas not occurring in the branch and use inevitably the cut rule to make the whole construction work, the latter being a kind of Henkin construction. For this reason we prefer the much simpler syntactic proof.

Theorem 4.17 (Completeness)**.** *If A is valid in CK or in one of its mentioned extensions, then $\vdash x : A$ is derivable in the respective SeqS system.*

Proof. If A is valid in CK or in one of its mentioned extensions, then $\vdash_S A$ is a theorem in the corresponding axiomatization by the completeness of the axioms (Theorem 2.2). We show that if $\vdash_S A$ is a theorem, then $\vdash x : A$ is derivable in SeqS. We must show that the axioms are derivable and that the set of derivable formulas is closed under (Modus Ponens), (RCEA), and (RCK). A derivation of axioms (**ID**), (**MP**), (**CS**) and (**CEM**) can be obtained from Examples 4.3, 4.4, 4.5 and 4.6 respectively; indeed, by Lemma 4.8, one can generalize these proofs to the case in which a propositional variable P is replaced by any formula A. Let us examine the other rules.
For (Modus Ponens), suppose that $\vdash x : A \rightarrow B$ and $\vdash x : A$ are derivable. We easily have that $x : A \rightarrow B, x : A \vdash x : B$ is derivable too. Since cut is admissible, by two cuts we obtain $\vdash x : B$, as follows:

$$\dfrac{\dfrac{x : A \rightarrow B, x : A \vdash x : B \qquad \vdash x : A \rightarrow B}{x : A \vdash x : B}\,(cut) \qquad \vdash x : A}{\vdash x : B}\,(cut)$$

For (RCEA), we have to show that if $A \leftrightarrow B$ is derivable, then also $(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ is so. The formula $A \leftrightarrow B$ is an abbreviation for $(A \rightarrow B) \wedge (B \rightarrow A)$. Suppose that $\vdash x : (A \rightarrow B) \wedge (B \rightarrow A)$ is derivable, then by Theorem 4.11 also $x : A \vdash x : B$ and $x : B \vdash x : A$ are derivable. We can derive $x : A \Rightarrow C \vdash x : B \Rightarrow C$ as follows: (the other half is symmetric).

$$\dfrac{\genfrac{}{}{0pt}{}{x : A \vdash x : B \qquad x : B \vdash x : A}{x : A \Rightarrow C, x \xrightarrow{B} y \vdash x \xrightarrow{A} y, y : C}\,(\mathbf{EQ}) \qquad x : A \Rightarrow C, x \xrightarrow{B} y, y : C \vdash y : C}{\genfrac{}{}{0pt}{}{x \xrightarrow{B} y, x : A \Rightarrow C \vdash y : C}{x : A \Rightarrow C \vdash x : B \Rightarrow C}\,(\Rightarrow \mathbf{R})}\,(\Rightarrow \mathbf{L})$$

For (RCK), supposed that (1) $\vdash x : B_1 \wedge B_2 \cdots \wedge B_n \rightarrow C$ is derivable, by Theorem 4.11 and Lemma 4.9, also $y : B_1, \ldots, y : B_n \vdash y : C$ is derivable. Then we have (we omit side formulas in $x \xrightarrow{A} y \vdash x \xrightarrow{A} y$):

$$\dfrac{x \xrightarrow{A} y \vdash x \xrightarrow{A} y \qquad x : A \Rightarrow B_1, \ldots, x : A \Rightarrow B_n, y : B_1, \ldots, y : B_n \vdash y : C}{x \xrightarrow{A} y, x : A \Rightarrow B_1, \ldots, x : A \Rightarrow B_n, y : B_1, \ldots, y : B_{n-1} \vdash y : C}\,(\Rightarrow \mathbf{L})$$

$$\vdots$$

$$\dfrac{\dfrac{x \xrightarrow{A} y \vdash x \xrightarrow{A} y \qquad x \xrightarrow{A} y, x : A \Rightarrow B_1, \ldots, x : A \Rightarrow B_n, y : B_1 \vdash y : C}{x \xrightarrow{A} y, x : A \Rightarrow B_1, \ldots, x : A \Rightarrow B_n \vdash y : C}\,(\Rightarrow \mathbf{L})}{x : A \Rightarrow B_1, \ldots, x : A \Rightarrow B_n \vdash x : A \Rightarrow C}\,(\Rightarrow \mathbf{R})$$

■

4.4 Decidability and Complexity

In this section we analyze SeqS calculi in order to obtain a decision procedure for all conditional systems under consideration[5]. We first present some common properties, then we analyze separately systems CK{+ID}{+MP}, systems CEM{+ID} and systems CS*.

In general, cut-freeness alone does not ensure termination of proof search in a sequent calculus; the presence of labels and of the $(\Rightarrow \mathbf{L})$ rule, which increases the complexity of the sequent in a backward proof search, are potential causes of a non-terminating proof search. In this section we show that SeqS's rules introduce only a finite number of labels in a backward proof search, and that $(\Rightarrow \mathbf{L})$ can be applied in a controlled way: these conditions allow to describe a decision procedure for the corresponding logics. We also give explicit complexity bounds for our systems.

[5] Obviously, we restrict our concern to all cut-free systems, i.e. all SeqS systems except SeqCEM+MP*.

As a first step, we show that it is useless to apply ($\Rightarrow$ **L**) on $x : A \Rightarrow B$ by introducing (looking backward) the same transition formula $x \xrightarrow{A} y$ *more than once in each branch of a proof tree.* More in detail, we have the following:

Lemma 4.18 (Controlled use of ($\Rightarrow$ **L**))**.** *If* $\Gamma \vdash \Delta$ *is derivable, then there is a proof of it which does not contain more than one application of (*$\Rightarrow$ **L***) (looking backward) on* $x : A \Rightarrow B$ *introducing the same transition formula* $x \xrightarrow{A} y$ *in each branch.*

Proof. Consider a derivation of $\Gamma \vdash \Delta$ in which ($\Rightarrow$ **L**) is applied to $x : A \Rightarrow B$ with a transition $x \xrightarrow{A} y$ more than once in a branch; in particular, consider the two highest[6] applications. We have the following situation:

$$\begin{array}{c}
\dfrac{\overset{\Pi_a}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y} \qquad \overset{\Pi_b}{\Gamma_1, x : A \Rightarrow B, y : B \vdash \Delta_1}}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1}\ (\Rightarrow \mathbf{L}) \\
\vdots \\
\Gamma \vdash \Delta
\end{array}$$

and in Π_a or in Π_b the rule ($\Rightarrow$ **L**) is applied (looking backward) to $x : A \Rightarrow B$ by using $x \xrightarrow{A} y$. If the highest application is in Π_a we have:

$$\dfrac{\begin{array}{c}
\dfrac{\Gamma_2, x : A \Rightarrow B \vdash \Delta_2, x \xrightarrow{A} y \qquad \Gamma_2, x : A \Rightarrow B, y : B \vdash \Delta_2}{\Gamma_2, x : A \Rightarrow B \vdash \Delta_2}\ (\Rightarrow \mathbf{L}) \\
\vdots \\
\Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y
\end{array} \qquad \overset{\Pi_b}{\Gamma_1, x : A \Rightarrow B, y : B \vdash \Delta_1}}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1}\ (\Rightarrow \mathbf{L})$$

The application of ($\Rightarrow$ **L**) in the left branch can be permuted over the other rules in Π_a (remember that ($\Rightarrow$ **L**) is invertible, see Theorem 4.11); we have the following proof tree ($\Pi_a^{'}$ is Π_a after the permutation):

$$\dfrac{\dfrac{\overset{\Pi_a^{'}}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y, x \xrightarrow{A} y} \qquad \ldots y : B \vdash \ldots}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y}\ (\Rightarrow \mathbf{L}) \qquad \overset{\Pi_b}{\Gamma_1, x : A \Rightarrow B, y : B \vdash \Delta_1}}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1}\ (\Rightarrow \mathbf{L})$$

By contraction (Theorem 4.12), we have a proof $\Pi_a^{''}$ of $\Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y$, which does not contain any application of ($\Rightarrow$ **L**) on $x : A \Rightarrow B$ introducing the same transition $x \xrightarrow{A} y$ (remember that contraction is rule-preserving admissible) and then we have the following proof:

[6] The applications having the greatest distance from the root $\Gamma \vdash \Delta$.

$$\begin{array}{c}
\dfrac{\begin{array}{cc} \Pi_a^{''} & \Pi_b \\ \Gamma_1, x : A \Rightarrow B \vdash \Delta_1, x \xrightarrow{A} y & \Gamma_1, x : A \Rightarrow B, y : B \vdash \Delta_1 \end{array}}{\Gamma_1, x : A \Rightarrow B \vdash \Delta_1} \; (\Rightarrow \mathbf{L}) \\
\vdots \\
\Gamma \vdash \Delta
\end{array}$$

If the highest application is in Π_b the proof is similar and left to the reader.

■

From now on, we analyze separately the decidability of systems CK{+ID}{+MP}, CEM{+ID} and CS*.

4.4.1 Termination and Complexity for CK{+ID}{+MP}

In this subsection we prove some properties characterizing calculi SeqS for conditional logics CK{+ID}{+MP}, in order to give a decision procedure for these systems. From now on we only refer to calculi for these systems unless stated otherwise.

First of all, observe that the calculi are characterized by the following property:

Theorem 4.19 (Property of right-transition formulas)**.** *Let the sequent*

$$\Gamma \vdash \Delta, x \xrightarrow{A} y$$

with $x \neq y$, be derivable, then one of the following sequents:

1. $\Gamma \vdash \Delta$
2. $x \xrightarrow{F} y \vdash x \xrightarrow{A} y$, *where* $x \xrightarrow{F} y \in \Gamma$

is also derivable.

Proof. (**EQ**) is the only rule which operates, considering a backward proof search, on a transition formula on the right hand side (consequent) of a sequent. Thus, we have to consider three cases, analyzing the proof tree of $\Gamma \vdash \Delta, x \xrightarrow{A} y$:

- $\Gamma \vdash \Delta, x \xrightarrow{A} y$ is an axiom: we have to consider two subcases:
  - ⋆ an atom $u : P$ occurs in both Γ and Δ, therefore $\Gamma \vdash \Delta$ is derivable;
  - ⋆ $w : \bot \in \Gamma$, then $\Gamma \vdash \Delta$ is derivable;
- $x \xrightarrow{A} y$ is never principal in the derivation. In this case, $\Gamma \vdash \Delta$ is derivable, since we can remove an occurrence of $x \xrightarrow{A} y$ from every sequent descending (looking forward) from $\Gamma_1 \vdash \Delta_1, x \xrightarrow{A} y$;
- $x \xrightarrow{A} y$ is introduced (looking forward) by the (**EQ**) rule: in this case, another transition $x \xrightarrow{F} y$ *must* be in Γ, in order to apply (**EQ**). To see this, observe that the only rule that could introduce a transition formula (looking backward) in the antecedent of a sequent is ($\Rightarrow$ **R**), but it can only introduce a transition of the form $x \xrightarrow{F} z$, where *z does not occur in that sequent* (it is a *new* label),

thus it cannot introduce the transition $x \xrightarrow{F} y$.
The (**EQ**) rule is only applied to transition formulas:

$$\begin{array}{c} u : F \vdash u : A \qquad u : A \vdash u : F \\ x \xrightarrow{F} y \vdash x \xrightarrow{A} y \end{array} \ (\mathbf{EQ})$$

therefore we can conclude that $x \xrightarrow{F} y \vdash x \xrightarrow{A} y$ is derivable.

■

Notice that this theorem holds for all the systems under consideration, but only if $x \neq y$. In systems with MP, considering a backward proof search, the (**MP**) rule operates on transitions in the consequent, although only on transitions of the form $x \xrightarrow{A} x$. In this case the theorem does not hold, as shown by the following counterexample:

$$\begin{array}{c} x : A \vdash x \xrightarrow{A} x, x : A, x : B \\ x : A \vdash x \xrightarrow{A} x, x : B \end{array} \ (\mathbf{MP})$$

for A and B arbitrary. The sequent $x : A \vdash x \xrightarrow{A} x, x : B$ is derivable in SeqMP, but $x : A \vdash x : B$ is not derivable in this system and the second condition is not applicable (no transition formula occurs in the antecedent). The first hypothesis of the theorem ($x \neq y$) excludes this situation.

In order to control the application of ($\Rightarrow$ **L**) we show that it is useless to apply (backward) the ($\Rightarrow$ **L**) rule on $x : A \Rightarrow B$ by introducing a transition $x \xrightarrow{A} y$ if no $x \xrightarrow{A'} y$ belongs to the left-hand side of the sequent, since there will be no way to prove that transition. This property is stated by the following:

Lemma 4.20 (Controlled use of ($\Rightarrow$ **L**) for CK{+ID}{+MP})**.** *SeqS calculi for CK{+ID}{+MP} are complete even if the ($\Rightarrow$ **L**) rule is applied as follows:*

$$\dfrac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow \mathbf{L})$$

by choosing a label y such that:

- *(for CK{+ID}) there is a transition $x \xrightarrow{A'} y \in \Gamma$;*
- *(for CK+MP{+ID}) there is a transition $x \xrightarrow{A'} y \in \Gamma$ or $y = x$.*

Proof. Let us consider a derivation where ($\Rightarrow$ **L**) is applied (backward) to $\Gamma, x : A \Rightarrow B \vdash \Delta$ by introducing a transition $x \xrightarrow{A} y$ such that no transitions of the form $x \xrightarrow{A'} y$ belong to Γ and $y \neq x$. The left premise of the rule is $\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y$: by Theorem 4.19 we have that $\Gamma, x : A \Rightarrow B \vdash \Delta$ is derivable, then the application of ($\Rightarrow$ **L**) under consideration is useless.

■

In order to get a decision procedure for these logics we need to control the application of (**ID**) and (**MP**), whose premises have a higher complexity than the respective conclusions. We prove that it is useless to apply (**ID**) and (**MP**) more than once on the same transition in each derivation branch, as stated by the following lemmas:

Lemma 4.21 (Controlled use of (**ID**)). *It is useless to apply* (**ID**) *on the same transition* $x \xrightarrow{A} y$ *more than once in a backward proof search in each branch of a derivation.*

Proof. Consider a proof where (**ID**) is applied more than once on the same transition in a derivation and consider the two highest applications: since (**ID**) is invertible (Theorem 4.11), we can consider, without loss of generality, that the two applications of (**ID**) are consecutive, as follows:

$$\dfrac{\dfrac{(1)\Gamma, x \xrightarrow{A} y, y : A, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta}\,(\mathbf{ID})}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{ID})$$

From (1) we can find a derivation of $(1^{'})\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta$ by contraction, and this derivation does not have any application of (**ID**) having $x \xrightarrow{A} y$ as a principal formula (remember that contraction is rule-preserving admissible). Thus, we can remove one application of (**ID**) as follows:

$$\dfrac{(1^{'})\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{ID})$$

■

Lemma 4.22 (Controlled use of (**MP**)). *It is useless to apply* (**MP**) *on the same transition* $x \xrightarrow{A} x$ *more than once in a backward proof search in each branch of a derivation.*

Proof. The proof is similar to the proof of Lemma 4.21 and left to the reader.

Now we have all the elements to prove the decidability of systems for CK{+ID} {+MP}:

Theorem 4.23 (Termination for CK{+MP}{+ID}). *Systems SeqCK, SeqID, SeqMP and SeqID+MP ensure termination.*

Proof. In all rules the premises have a smaller complexity than the conclusion, except ($\Rightarrow$ **L**), (**ID**) and (**MP**). However, Lemma 4.18 guarantees that ($\Rightarrow$ **L**) can be applied in a controlled way, i.e. one needs to apply ($\Rightarrow$ **L**) only once on a formula $x : A \Rightarrow B$ with the same transition $x \xrightarrow{A} y$ in each branch. Moreover, considering a derivation of a sequent $\Gamma, x : A \Rightarrow B \vdash \Delta$, the number of applications of ($\Rightarrow$ **L**) on $x : A \Rightarrow B$ is bounded by the cardinality of $\mathcal{B} = \{x \xrightarrow{A} y \mid x \xrightarrow{A^{'}} y \in \Gamma\}$[7] by Lemma 4.20. The

[7] In systems allowing the (**MP**) rule the number of applications of ($\Rightarrow$ **L**) is bounded by the cardinality of $\mathcal{B} = \{x \xrightarrow{A} y \mid x \xrightarrow{A^{'}} y \in \Gamma\} \cup \{x \xrightarrow{A} x\}$.

number of different y such that $x \xrightarrow{A} y \in \mathcal{B}$ is finite, since labels are only introduced by conditional formulas occurring negatively in the initial sequent, which are finite.

Lemmas 4.21 and 4.22 guarantee that we only need a finite number of applications of (**ID**) and (**MP**) in a backward proof search. Moreover, observe that the rules are analytic, so that the premises contains only (labelled) subformulas of the formulas in the conclusion. In the search of a proof of $\vdash x_0 : D$, with $\mid D \mid= n$, new labels are introduced only by conditional subformulas occurring negatively in D.

The number of different labels occurring in a proof is $O(n)$, and the length of each branch of a proof tree is bounded by $O(n^2)$.

■

This itself gives decidability:

Theorem 4.24 (CK{+ID}{+MP} decidability)**.** *Logic CK{+ID} is decidable.*

Proof. We just observe that there is only a finite number of derivations to check of a given sequent $\vdash x_0 : D$, as both the length of a proof and the number of labelled formulas which may occur in it is finite.

■

We conclude this subsection by giving an explicit space complexity bound for CK{+ID} {+MP}. As usual, a proof may have an exponential size because of the branching introduced by the rules. However we can obtain a much sharper space complexity bound since we do not need to store the whole proof, but only a sequent at a time plus additional information to carry on the proof search; this standard technique is similar to the one adopted in [Hud93] and [Vig00].

Theorem 4.25 (Space complexity of CK{+ID}{+MP})**.** *Provability in CK{+ID} {+MP} is decidable in $O(n^2 \log n)$ space.*

Proof. First, we observe that, in searching a proof, there are two kinds of branching to consider: AND-branching caused by the rules with multiple premises and OR-branching (backtracking points in a depth first search) caused by the choice of the rule to apply.

We store only one sequent at a time and maintain a stack containing information sufficient to reconstruct the branching points of both types. Each stack entry contains the principal formula (either a world formula $x : B$, or a transition formula $x \xrightarrow{B} y$), the name of the rule applied and an index which allows to reconstruct the other branches on return to the branching points. The stack entries represent thus backtracking points and the index within the entry allows one to reconstruct both the AND branching and to check whether there are alternatives to explore (OR branching). The working sequent on a return point is recreated by replaying the stack entries from the bottom of the stack using the information in the index (for instance in the case of ($\Rightarrow$ **L**) applied to the principal formula $x : A \Rightarrow B$, the index will indicate which premise-first or second-we have to expand and the label y involved in the transition formula $x \xrightarrow{A} y$).

A proof begins with the end sequent $\vdash x_0 : D$ and the empty stack. Each rule application generates a new sequent and extends the stack. If the current sequent is an axiom we pop the stack until we find an AND branching point to be expanded. If there are not, the end sequent $\vdash x_0 : D$ is provable and we have finished. If the current sequent is not an axiom and no rule can be applied to it, we pop the stack entries and we continue at the first available entry with some alternative left (a backtracking point). If there are no such entries, the end sequent is not provable.

The entire process must terminate since: (i) the depth of the stack is bounded by the length of a branch proof, thus it is $O(n^2)$, where $\mid D \mid= n$, (ii) the branching is bounded by the number of rules, the number of premises of any rule and the number of labelled formulas occurring in one sequent, the last being $O(n^2)$.

To evaluate the space requirement, we have that each subformula of the initial labelled formula can be represented by a positional index into the initial labelled formula, which requires $O(\log n)$ bits. Moreover, also each label can be represented by $O(\log n)$ bits. Thus, to store the working sequent we need $O(n^2 \ \log n)$ space, since there may occur $O(n^2)$ labelled subformulas. Similarly, each stack entry requires $O(\log n)$ bits, as the name of the rule requires constant space and the index $O(\log n)$ bits. Having depth $O(n^2)$, to store the whole stack requires $O(n^2 \ \log n)$ space. Thus we obtain that provability in CK{+ID}{+MP} is decidable in $O(n^2 \ \log n)$ space.

■

4.4.2 Termination and Complexity for CK+CEM{+ID}

In this subsection we analyze sequent calculi containing (**CEM**). In order to show that SeqCEM{+ID} ensure termination we proceed in a similar manner as we made in the previous subsection; in particular, we have to show that both (**CEM**) and ($\Rightarrow$ **L**) rules can be applied in a controlled way. The principal formula of these rules is maintained in their premises, and this is a potential cause of non termination in a backward proof search. However, we prove that the number of applications of both (**CEM**) and ($\Rightarrow$ **L**) is finite, and this gives the decidability.

Lemma 4.26 (Controlled use of (**CEM**) (Part 1))**.** *SeqCEM{+ID} are complete even if the (***CEM***) rule is applied with the following restrictions:*

$$\frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z \qquad (\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\ (\mathbf{CEM})$$

1. $y \neq z$;

2. there exists a transition $x \xrightarrow{A'} z \in \Gamma$.

Proof. Consider the transition $x \xrightarrow{A} z$ introduced (backward) by the (**CEM**) rule in its left premise. If it is introduced by weakening, it can be removed and we are done (the application of (**CEM**) is useless). Otherwise, it derives (forward) from an application of (**EQ**) or it is used in the derivation of the right premise of another application of (**CEM**) by the identification of labels (for instance (**CEM**) is also applied to $x \xrightarrow{B} y$ and in the right premise y and z are identified with a new label u). In the first case, a transition $x \xrightarrow{A'} z$ must belong to Γ and the restriction 2. is satisfied. In the second case, we observe that $x \xrightarrow{A} z$ *still appears in the right-hand side of the left premise of the rule*, therefore it can be derived by (**EQ**) or used by

(**CEM**) in the derivation of the right premise, and so on. Obviously, given a proof tree of $\Gamma' \vdash \Delta, x \xrightarrow{A} z$, we can repeat this reasoning on each left premise of an application of (**CEM**) using $x \xrightarrow{A} z$ in its right derivation, until we find that $x \xrightarrow{A} z$ is derived by an application of (**EQ**) or by weakening, since the proof tree is finite, as shown below (we can assume, without loss of generality, that all the applications of (**CEM**) are consecutive, since (**CEM**) is invertible and then we can permute it over the other rules):

$$
\begin{array}{c}
\begin{array}{ll}
\quad\Pi & \\
\Gamma' \vdash \Delta_n, x \xrightarrow{A} z & \quad \dots
\end{array} \\
\vdots \\
\dfrac{\Gamma' \vdash \Delta_2, x \xrightarrow{A} z \qquad \dots}{\Gamma' \vdash \Delta_1, x \xrightarrow{A} z \qquad \dots}\ (\mathbf{CEM}) \\
\ (\mathbf{CEM}) \\
\Gamma' \vdash \Delta, x \xrightarrow{A} z
\end{array}
$$

In Π the transition $x \xrightarrow{A} z$ can only be introduced by weakening or by an application of (**EQ**) with a transition $x \xrightarrow{A'} z$ in the left-hand side of a sequent. In the first case, all instances of $x \xrightarrow{A} z$ can be removed; in the second case, we can conclude that the transition $x \xrightarrow{A'} z$ belongs to Γ', since we can reason as in the proof of Theorem 4.19: ($\Rightarrow$ **R**) is the only rule introducing (looking backward) a transition $x \xrightarrow{A'} z$ in the left-hand side of a sequent; moreover, z is a new label, then it is not possible that $x \xrightarrow{A'} z$ is introduced in Π, since z already occurs in all sequents; thus, $x \xrightarrow{A'} z \in \Gamma$.

Notice that the restriction 1. is the initial restriction given in SeqCEM{+ID} in order to avoid a looping application of (**CEM**).

■

Similarly to (**CEM**) we can control the application of ($\Rightarrow$ **L**) as stated by the following:

Lemma 4.27 (Controlled use of ($\Rightarrow$ **L**) for CEM{+ID})**.** *SeqCEM{+ID} is complete even if the (*$\Rightarrow$ **L***) rule is applied as follows:*

$$
\dfrac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow \mathbf{L})
$$

by choosing a label y such that there is a transition $x \xrightarrow{A'} y \in \Gamma$.

Proof. The proof is similar to the proof of Lemma 4.26 and then left to the reader.

To prove that SeqCEM{+ID} ensure termination we also need the following:

Lemma 4.28 (Controlled use of (**CEM**) (Part 2))**.** *It is useless to apply* (**CEM**) *to $x \xrightarrow{A} y$ by introducing the same transition $x \xrightarrow{A} z$ in the left premise of the rule more than once in each branch of a backward proof search.*

Proof. Consider a proof where (**CEM**) is applied to $x \xrightarrow{A} y$ more than once by introducing the same transition $x \xrightarrow{A} z$ in a branch; consider the two highest applications:

since (**CEM**) is invertible, it permutes over the other rules, then we can consider, without loss of generality, the following proof:

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z, x \xrightarrow{A} z} \quad (\Gamma, x \xrightarrow{A} y \vdash \dots)[y, z/u]}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z}\,(\mathbf{CEM}) \qquad \overset{\Pi_2}{(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y, z/u]}}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CEM})$$

By contraction, one can find a proof $\Pi_1^{'}$ of the sequent $\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z$, then we can conclude as follows, obtaining a proof where the upper application of (**CEM**) has been removed:

$$\dfrac{\overset{\Pi_1^{'}}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z} \qquad \overset{\Pi_2}{(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y, z/u]}}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CEM})$$

■

By the above Lemmas 4.26, 4.27 and 4.28 we prove the decidability of CK+CEM{+ID}:

Theorem 4.29 (Termination for CK+CEM{+ID}). *Systems SeqCEM{+ID} ensure termination.*

Proof. We proceed as we made for systems CK{+ID}{+MP}. In particular, one can control the application of (⇒ **L**) and (**CEM**) by Lemmas 4.26, 4.28, 4.27 and 4.18.

■

As in the case of CK{+ID}{+MP}, this itself gives decidability:

Theorem 4.30 (Decidability of CK+CEM{+ID}). *Logic CK+CEM{+ID} is decidable.*

We can easily extend results for space complexity given in the previous subsection to systems allowing (**CEM**):

Theorem 4.31 (Space complexity of CK+CEM{+ID}). *Provability in CK+CEM{+ID} is decidable in $O(n^2 \log n)$ space.*

4.4.3 Termination and Complexity for CK+CS*

As we made in the previous subsections, we have to show that one can apply the (⇒ **L**) rule in a controlled way. We have also to show that (**CS**) can be controlled too.

However, in these systems we have a problem. Consider the following derivation:

Example 4.32.

$$\begin{array}{c}
\dfrac{x \xrightarrow{\top} y \vdash x \xrightarrow{\top} y \qquad \dfrac{\dfrac{\dfrac{\ldots x \xrightarrow{\top} y, y \xrightarrow{\top} z \vdash \ldots, x \xrightarrow{\top} z \qquad \overset{\Pi}{x : \top \Rightarrow (\neg(\top \Rightarrow A)), \ldots, z : \neg(\top \Rightarrow A) \vdash y : A, z : A}}{x : \top \Rightarrow (\neg(\top \Rightarrow A)), x \xrightarrow{\top} y, y \xrightarrow{\top} z \vdash y : A, z : A}\,(\Rightarrow \mathbf{L})}{x : \top \Rightarrow (\neg(\top \Rightarrow A)), x \xrightarrow{\top} y, \vdash y : A, y : \top \Rightarrow A}\,(\Rightarrow \mathbf{R})}{x : \top \Rightarrow (\neg(\top \Rightarrow A)), x \xrightarrow{\top} y, y : \neg(\top \Rightarrow A) \vdash y : A}\,(\neg \mathbf{L})}{x : \top \Rightarrow (\neg(\top \Rightarrow A)), x \xrightarrow{\top} y \vdash y : A}\,(\Rightarrow \mathbf{L}) \\
(\Rightarrow \mathbf{R}) \\
x : \top \Rightarrow (\neg(\top \Rightarrow A)) \vdash x : \top \Rightarrow A
\end{array}$$

In the above derivation there is a loop by the combination of the following facts:

1. the conditional formulas $x : \top \Rightarrow A$, $y : \top \Rightarrow A$, ..., generate *new* labels in the proof tree;

2. one may have *transitive transitions*, as $x \xrightarrow{\top} y, y \xrightarrow{\top} z \vdash ..., x \xrightarrow{\top} z$.

More generally, we can have sequents of the form $\Gamma, x_0 \xrightarrow{A_1} x_1, x_1 \xrightarrow{A_2} x_2, ..., x_{n-1} \xrightarrow{A_n} x_n \vdash \Delta, x_0 \xrightarrow{A} x_n$ derived by applying the (**CS**) rule. Intuitively, the reason is that an application of (**CS**) on $x_{i-1} \xrightarrow{A_i} x_i$ has the effect of identifying labels x_{i-1} and x_i, therefore several backwards applications of this rule lead to a sequent of the form $\Gamma^{'}, u \xrightarrow{A_n} x_n \vdash \Delta^{'}, u \xrightarrow{A} x_n$, which can be derived for instance by (**EQ**).

We can remedy to this problem by restricting the application of ($\Rightarrow$ **L**) on $\Gamma, x : A \Rightarrow B \vdash \Delta$ by using only transitions $x \xrightarrow{A} y$ such that $x \xrightarrow{A^{'}} y \in \Gamma$. The reason why we can control the application of ($\Rightarrow$ **L**) can be explained as follows: one may have transitive transitions *since* (**CS**) *identifies two labels in its right premise.* Indeed, to prove $x \xrightarrow{A} y, y \xrightarrow{A} z \vdash x \xrightarrow{A} z$ one can apply (**CS**) on $x \xrightarrow{A} y$: the right premise $u \xrightarrow{A} u, u \xrightarrow{A} z \vdash u \xrightarrow{A} z$ is derivable by the identification of labels x and y with u. However, the transition $x \xrightarrow{A} z$ is maintained in the left premise, where it can only be introduced by an application of (**EQ**) or by weakening. The intuition is that if one needs to propagate a conditional $x : A \Rightarrow B$ from x to y, and then to z by an application of (**CS**), where (**CS**) has the effect of identifying x and y, then one can *first* identify labels x and y with u by (**CS**), and *then* propagate the conditional $u : A \Rightarrow B$ from u to z by an application of ($\Rightarrow$ **L**).

Lemma 4.33 (Controlled use of ($\Rightarrow$ **L**) for CS*)**.** *SeqCS* are complete even if the (*$\Rightarrow$ **L***) rule is applied as follows:*

$$\dfrac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta}\,(\Rightarrow \mathbf{L})$$

by choosing a label y *such that:*

- *(for systems without MP) there is a transition* $x \xrightarrow{A^{'}} y \in \Gamma$*;*

- *(for systems with MP) there is a transition* $x \xrightarrow{A^{'}} y \in \Gamma$ *or* $y = x$*.*

Proof. Consider a proof tree where ($\Rightarrow$ **L**) is applied introducing transitions by transitivity; the situation is as follows (as usual, we denote with $\Sigma(u)$ the substitution $\Sigma[x/u, y/u]$):

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma, x \xrightarrow{A^{'}} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} z, x : A'} \qquad \overset{\Pi_2}{\Gamma(u), u \xrightarrow{A'} u \ldots \vdash \Delta, u \xrightarrow{A} z}}{\Gamma, x \xrightarrow{A'} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} z}\,(\mathbf{CS}) \qquad \overset{\Pi_3}{\Gamma, x \xrightarrow{A'} y, \ldots, z : B \vdash \Delta}}{\Gamma, x \xrightarrow{A'} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta}\,(\Rightarrow \mathbf{L})$$

We show that there exists a proof tree of $\Gamma, x \xrightarrow{A'} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta$ where $(\Rightarrow \mathbf{L})$ is applied without introducing transitions by transitivity. If $x \xrightarrow{A} z$ derives in Π_1 by an application of $(\mathbf{EQ})$, then a transition $x \xrightarrow{C} z$ belongs to the left-hand side of the sequent, therefore $(\Rightarrow \mathbf{L})$ is applied respecting the restriction stated by this Lemma. Otherwise, $x \xrightarrow{A} z$ can be removed in Π_1, since it is introduced by weakening[8]. Therefore, there is a proof $\Pi_1^{'}$ of the sequent $\Gamma, x \xrightarrow{A^{'}} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta, x : A'$. Moreover, by applying the label substitution (Lemma 4.9) to the sequent derived by proof Π_3 we can obtain a proof $\Pi_3^{'}$ of $\Gamma(u), u \xrightarrow{A'} u, u \xrightarrow{A''} z, u : A \Rightarrow B, z : B \vdash \Delta(u)$. We can conclude by applying first the $(\mathbf{CS})$ rule (to identify labels x and y with u) and then by propagating the conditional formula from u to z, as follows:

$$\dfrac{\overset{\Pi_1^{'}}{\Gamma, x \xrightarrow{A^{'}} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta, x : A'} \qquad \dfrac{\overset{\Pi_2}{\Gamma(u), u \xrightarrow{A'} u \ldots \vdash \Delta, u \xrightarrow{A} z} \qquad \overset{\Pi_3^{'}}{\Gamma(u), u \xrightarrow{A'} u, \ldots, z : B \vdash \Delta(u)}}{\Gamma(u), u \xrightarrow{A'} u, u \xrightarrow{A''} z, u : A \Rightarrow B \vdash \Delta(u)}\,(\Rightarrow \mathbf{L})}{\Gamma, x \xrightarrow{A'} y, y \xrightarrow{A''} z, x : A \Rightarrow B \vdash \Delta}\,(\mathbf{CS})$$

where the $(\Rightarrow \mathbf{L})$ rule has been applied respecting the restriction of this lemma, i.e. by introducing (backward) a transition $u \xrightarrow{A} z$ such that $u \xrightarrow{A^{''}} z$ belongs to the left-hand side of the sequent. We proceed in the same manner if we have a proof where $(\mathbf{CS})$ is applied on $y \xrightarrow{A''} z$ and also when the transition $x_0 \xrightarrow{A} x_n$ is used to apply $(\Rightarrow \mathbf{L})$ on $\Gamma, x_0 \xrightarrow{A_1} x_1, x_1 \xrightarrow{A_2} x_2, ..., x_{n-1} \xrightarrow{A_n} x_n$.

■

As in the case of $(\mathbf{CEM})$ we need to show that $(\mathbf{CS})$ can be applied in a controlled way, in order to show that SeqCS* ensure termination.

Lemma 4.34 (Controlled use of $(\mathbf{CS})$)**.** *It is useless to apply* $(\mathbf{CS})$ *on the same transition* $x \xrightarrow{A} y$ *more than once in each branch of a backward proof search.*

Proof. Consider a branch where $(\mathbf{CS})$ is applied more than once on $x \xrightarrow{A} y$ and consider the two highest applications; without loss of generality, we can consider the following proof since $(\mathbf{CS})$ is invertible (see Theorem 4.11, as usual we denote $\Sigma(u) = \Sigma[x/u, y/u]$):

[8] We can have several consecutive applications of $(\mathbf{CS})$. However, as in the case of $(\mathbf{CEM})$, $x \xrightarrow{A} z$ can be derived either by $(\mathbf{EQ})$ or by weakening since it is maintained in the consequent of the left premise of $(\mathbf{CS})$.

$$\dfrac{\dfrac{(1)\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A, x : A \qquad \Gamma(u), u \xrightarrow{A} u \vdash \Delta(u), u : A}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A}\,(\mathbf{CS}) \qquad (2)\Gamma(u), u \xrightarrow{A} u \vdash \Delta(u)}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CS})$$

By Theorem 4.12 we can find a proof of $(1^{'})\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A$, thus we conclude as follows:

$$\dfrac{(1^{'})\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A \qquad (2)\Gamma(u), u \xrightarrow{A} u \vdash \Delta(u)}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\,(\mathbf{CS})$$

We have found a derivation of the initial sequent in which a useless application of (**CS**) has been removed.

■

Now we have all the elements to prove the following:

Theorem 4.35 (Termination for CK+CS*)**.** *Systems SeqCS* ensure termination.*

Proof. One can control the application of ($\Rightarrow$ **L**) by Lemma 4.33 and of (**CS**) by Lemma 4.34. For systems with (**ID**) and/or (**MP**), one can control the application of these rules since Lemmas 4.21 and 4.22 hold in systems with (**CS**) too. In the case of SeqCEM+CS* just observe that one can control the application of (**CEM**) in this way: one needs to apply (**CEM**) on $\Gamma, x \xrightarrow{A} y \vdash \Delta$ by using a transition $x \xrightarrow{A} z$ (i.e. premises are $\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A} z$ and $(\Gamma, x \xrightarrow{A} y \vdash \Delta)[y/u, z/u]$) such that $x \xrightarrow{A^{'}} z \in \Gamma$, since Lemma 4.26 holds in these systems too. Moreover, one needs to apply (**CEM**) at most once by using the same transition $x \xrightarrow{A} z$ in each branch, since we can easily observe that Lemma 4.28 holds in these systems[9]. The number of applications of ($\Rightarrow$ **L**) and (**CEM**) is finite, since the number of transitions introduced by conditionals occurring negatively in the initial sequent of a backward proof search is finite.

■

As in the previous cases, this itself gives decidability:

Theorem 4.36 (Decidability of CK+CS*)**.** *Logics CK+CS* are decidable.*

We conclude by giving an explicit space complexity bound:

Theorem 4.37 (Space complexity of CK+CS*)**.** *Provability in CK+CS* is decidable in $O(n^2 \log n)$ space.*

[9] We can repeat the same proof of the theorem even if we have (**CS**), since (**CEM**) and (**CS**) are both invertible.

$$(\mathbf{ID})\ \frac{\Gamma, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y \vdash \Delta} \qquad\qquad (\mathbf{MP})\ \frac{\Gamma \vdash x : A, \Delta}{\Gamma \vdash x \xrightarrow{A} x, \Delta}$$

Figure 4.4: Rules (**ID**) and (**MP**) reformulated for CK{+ID}{+MP}.

4.5 Refinements and Other Properties for CK{+ID}{+MP}

In this section we restrict our concern to the calculi for CK{+ID}{+MP} and we get a better terminating calculus and complexity bound for CK{+ID}. Intuitively, we can do this because these systems enjoy a sort of *disjunction property* for conditional formulas: if $(A_1 \Rightarrow B_1) \vee (A_2 \Rightarrow B_2)$ is valid, then either $(A_1 \Rightarrow B_1)$ or $(A_2 \Rightarrow B_2)$ is valid too.

First of all we observe that we can reformulate the rules (**ID**) and (**MP**) with the non invertible ones presented in Figure 4.4 (details are given in [Poz03]).

Let us introduce the notion of *regular sequent.* Intuitively, regular sequents are those sequents whose set of transitions in the antecedent forms a *forest.* We assume that trees do not contain cycles in the vertexes and that a forest is a set of trees. As we show in Proposition 4.40 below, any sequent in a proof beginning with a sequent of the form $\vdash x_0 : D$, for an arbitrary formula D, is regular. For this reason, we will restrict our concern to regular sequents.

We define the multigraph $\mathcal{G}$ of the transition formulas in the antecedent of a sequent:

Definition 4.38 (Multigraph of transitions $\mathcal{G}$)**.** *Given a sequent* $\Gamma \vdash \Delta$*, where* $\Gamma = \Gamma', T$ *and* T *is the multiset of transition formulas and* Γ' *does not contain transition formulas, we define the multigraph* $\mathcal{G} =< V, E >$ *associated to* $\Gamma \vdash \Delta$ *with vertexes* V *and edges* E*.* V *is the set of labels occurring in* $\Gamma \vdash \Delta$ *and* $< x, y >\in E$ *whenever* $x \xrightarrow{F} y \in T$*.*

Definition 4.39 (Regular sequent)**.** *A sequent* $\Gamma \vdash \Delta$ *is called regular if its associated multigraph of transitions* $\mathcal{G}$ *is a forest.*

The graph of transitions of regular sequents forms a forest, as shown in Figure 4.5.

As mentioned above, we can always restrict our concern to regular sequents, since we have the following Proposition[10]:

Proposition 4.40 (Proofs with regular sequents)**.** *Every proof tree with a sequent* $\vdash x_0 : D$ *as root and obtained by applying backward SeqS's rules contains only regular sequents.*

For technical reasons we introduce the following definition:

[10] Notice that this theorem does not hold in systems with CEM or CS: indeed, if (**CEM**) or (**CS**) is applied (looking backward) to $\Gamma, x \xrightarrow{A} y \vdash \Delta$, then two labels are identified in the right premise, thus the resulting graph of transitions is not a forest. The proof of the proposition can be found in [OPS05], Theorem 5.3.

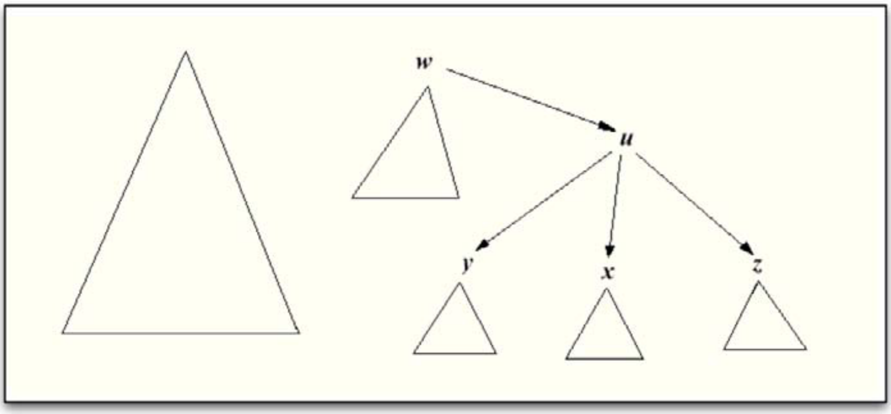


Figure 4.5: The forest $\mathcal{G}$ of a regular sequent.

Definition 4.41. *Given a forest $\mathcal{G}$ of transitions, let:*

- *$\mathcal{G}(k)$ be the tree of $\mathcal{G}$ with root k;*
- *r_k be the root of unique tree in $\mathcal{G}$ containing k.*

Observe that r_k may be the root of $\mathcal{G}(k)$.
Given a multiset of formulas Σ we define:

- *Σ_k° as the multiset of labelled formulas of Σ contained in $\mathcal{G}(k)$:*

$$\Sigma_k^\circ = \{u : F \in \Sigma \mid u \textit{ is a vertex of } \mathcal{G}(k)\} \cup$$
$$\cup \{u \xrightarrow{F} v \in \Sigma \mid \ v \textit{ is a vertex of } \mathcal{G}(k)\}$$

- *Σ_k^p as the multiset of labelled formulas of Σ contained on the path from r_k to k:*

$$\Sigma_k^p = \{u : F \in \Sigma \mid u \textit{ is on the path between } r_k \textit{ and } k\} \cup$$
$$\cup \{u \xrightarrow{F} v \in \Sigma \mid v \textit{ is on the path between } r_k \textit{ and } k\}$$

- *Σ_k^* as the multiset of labelled formulas of Σ contained in $\mathcal{G}(k)$ or on the path from r_k to k:*

$$\Sigma_k^* = \Sigma_k^\circ \ \cup \ \Sigma_k^p$$

Now we introduce the definition of x-branching formula. Intuitively, $\mathcal{B}(x, T)$ contains formulas that create a branching in x or in a predecessor of x according to T in a derivation of a sequent. For instance, consider the sequent $x : A, x : A \rightarrow B \vdash x : B$, obviously valid in CK. $x : A \rightarrow B$ is an x-branching formula, since it creates a branching in x in a derivation of the sequent:

$$\frac{x : A \vdash x : A, x : B \qquad x : A, x : B \vdash x : B}{x : A, x : A \rightarrow B \vdash x : B} \ (\rightarrow \mathbf{L})$$

$\mathcal{B}(x, T)$ also contains the conditionals $u : A \Rightarrow B$ such that $T \vdash u \xrightarrow{A} v$ and B creates a branching in x (i.e. $v = x$) or in a predecessor v of x.

Definition 4.42 (x-branching formulas)**.** *Given a multiset of transition formulas T, we define the set of x-branching formulas, denoted with $\mathcal{B}(x, T)$, as follows:*

- $x : A \to B \in \mathcal{B}(x,T)$;
- $u : A \to B \in \mathcal{B}(x,T)$ *if* $T \vdash u \xrightarrow{C} x$ *for some formula* C;
- $u : A \Rightarrow B \in \mathcal{B}(x,T)$ *if* $T \vdash u \xrightarrow{A} v$ *and* $v : B \in \mathcal{B}(x,T)$.

We also introduce the notion of x-branching sequent. Intuitively, we say that $\Gamma \vdash \Delta$ is x-branching if it contains an x-branching formula occurring positively in Γ or if it contains an x-branching formula occurring negatively in Δ. As an example, consider the following proof of $x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D) \vdash x : A \Rightarrow B, x : C \Rightarrow D$, valid in CK:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x \xrightarrow{A} y \vdash x \xrightarrow{A} y \qquad y : B \vdash y : B}{(*)x \xrightarrow{A} y, x \xrightarrow{C} z, x : A \Rightarrow B \vdash y : B, z : D, x : \bot}\;(\Rightarrow \mathbf{L})}{x \xrightarrow{A} y, x \xrightarrow{C} z \vdash y : B, z : D, x : (A \Rightarrow B) \to \bot}\;(\to \mathbf{R}) \qquad \dfrac{x \xrightarrow{C} z \vdash x \xrightarrow{C} z \qquad z : D \vdash z : D}{x \xrightarrow{A} y, x \xrightarrow{C} z, x : C \Rightarrow D \vdash y : B, z : D}\;(\Rightarrow \mathbf{L})}{x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D), x \xrightarrow{A} y, x \xrightarrow{C} z \vdash y : B, z : D}\;(\to \mathbf{L})}{x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D), x \xrightarrow{A} y \vdash y : B, x : C \Rightarrow D}\;(\Rightarrow \mathbf{R})}{x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D) \vdash x : A \Rightarrow B, x : C \Rightarrow D}\;(\Rightarrow \mathbf{R})$$

The initial sequent $x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D) \vdash x : A \Rightarrow B, x : C \Rightarrow D$ is x-branching by the formula $x : ((A \Rightarrow B) \to \bot) \to (C \Rightarrow D)$, which creates a branching on x in the derivation. The sequent $(*)x \xrightarrow{A} y, x \xrightarrow{C} z, x : A \Rightarrow B \vdash y : B, z : D, x : \bot$ is *not* x-branching, since no formula creates a branching in the backward proof search on x or on a path to x.

Since in systems containing (**ID**) a transition $u \xrightarrow{F} v$ in the antecedent can be derived, looking forward, from $v : F$ and $v : F$ can be x-branching, we impose that a sequent $\Gamma^{'}, u \xrightarrow{F} v \vdash \Delta$ is x-branching if $\Gamma^{'}, v : F \vdash \Delta$ is x-branching; by the same reason, in systems containing (**MP**) we impose that a sequent $\Gamma \vdash \Delta^{'}, u \xrightarrow{F} u$ is x-branching if $\Gamma \vdash \Delta^{'}, u : F$ is x-branching.
In systems containing (**MP**) we also impose that a sequent $\Gamma^{'}, w : A \Rightarrow B \vdash \Delta$ is x-branching if the sequent $\Gamma^{'} \vdash \Delta, w \xrightarrow{A} w$ is derivable and w is a predecessor of x (or $w = x$), since $w : A$ can introduce x-branching formula(s) in the sequent.

Definition 4.43 (x-branching sequents)**.** *Given a sequent* $\Gamma \vdash \Delta$*, we denote by* $\Gamma^{'}$ *the world formulas in* Γ *and by* T *the transition formulas in* Γ*, so that* $\Gamma{=}\Gamma^{'}, T$*. To define when a sequent* $\Gamma \vdash \Delta$ *is x-branching according to each system, we consider the following conditions:*

1. *a world formula* $u : F \in \mathcal{B}(x,T)$ *occurs positively in* Γ*;*
2. *a world formula* $u : F \in \mathcal{B}(x,T)$ *occurs negatively in* Δ*.*
3. $T{=}T^{'}, u \xrightarrow{F} v$ *and the sequent* $\Gamma^{'}, T^{'}, v : F \vdash \Delta$ *is x-branching;*
4. $u \xrightarrow{F} u \in \Delta$ *and the sequent* $\Gamma \vdash \Delta^{'}, u : F$ *is x-branching* ($\Delta = \Delta^{'}, u \xrightarrow{F} u$)*;*

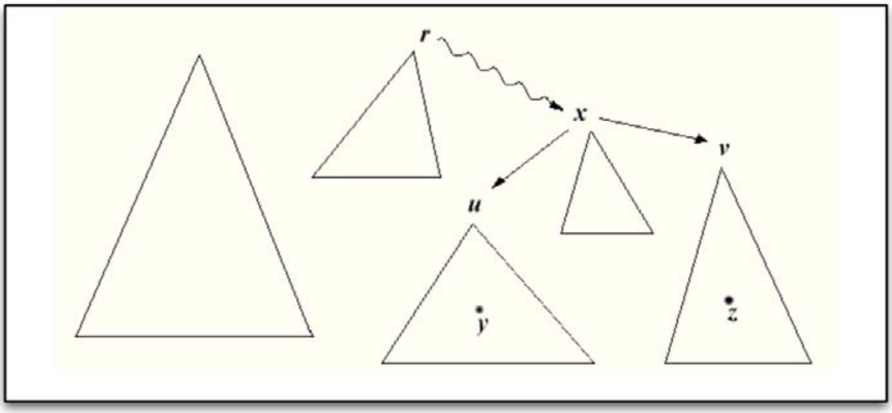


Figure 4.6: The forest $\mathcal{G}$ of transitions used to prove the disjunction property.

5. *a formula $w : A \Rightarrow B \in \Gamma$, w is a predecessor of x in the forest $\mathcal{G}$ of transitions or $w = x$ and $\Gamma'' \vdash \Delta, w \xrightarrow{A} w$ is derivable, where $\Gamma = \Gamma'', w : A \Rightarrow B$.*

We say that $\Gamma \vdash \Delta$ is x-branching for each system if the following combinations of the previous conditions hold:

- *CK: 1, 2*
- *CK+ID: 1, 2, 3*
- *CK+MP: 1, 2, 4, 5*
- *CK+MP+ID: 1, 2, 3, 4, 5*

As anticipated at the beginning of this section, the disjunction property *only* holds for sequents that are not x-branching. The reason is twofold: on the one hand, only the formulas on the path from x going backwards through the transition formulas (i.e. on the worlds $u_1 \xrightarrow{A_1} u_2 \xrightarrow{A_2} \ldots \xrightarrow{A_n} x$) can contribute to a proof of a formula with label x. This is proved by Proposition 4.44 below. On the other hand, no formula on that path can create a branching in the derivation. As an example, consider the x-branching sequent $x : ((A \Rightarrow B) \rightarrow \bot) \rightarrow (C \Rightarrow D) \vdash x : A \Rightarrow B, x : C \Rightarrow D$; it is valid in CK, but neither $x : ((A \Rightarrow B) \rightarrow \bot) \rightarrow (C \Rightarrow D) \vdash x : A \Rightarrow B$ nor $x : ((A \Rightarrow B) \rightarrow \bot) \rightarrow (C \Rightarrow D) \vdash x : C \Rightarrow D$ are valid.

To prove the disjunction property, we need to consider a more general setting; namely, we shall consider a sequent of the form $\Gamma \vdash \Delta, y : A, z : B$, whose forest of transitions has the form represented in Figure 4.6, i.e. it has one subtree with root u and another subtree with root v, with $u \neq v$; y is a member of the tree with root u and z is a member of the tree with root v; x is the father of u and v and the tree containing x has root r.

Now we have all the elements to prove the following proposition; intuitively, it says that if $\Gamma \vdash \Delta, y : A, z : B$ is derivable and its forest $\mathcal{G}$ has form as in Figure 4.6, then there is a derivation which involves (i) only the formulas whose labels are in $\mathcal{G}(u)$ or (ii) only the formulas whose labels are in $\mathcal{G}(v)$ or (iii) only the formulas whose labels are in the rest of the forest. This proposition is also crucial to prove the

Lemma 4.47 below, which leads to a better space complexity bound for CK{+ID}. More details are given in Proposition 5.12 in [OPS05].

Proposition 4.44. *Given a sequent $\Gamma \vdash \Delta, y : A, z : B$ and its forest of transitions $\mathcal{G}$, if it is derivable and has the following features:*

1. *$\mathcal{G}$ is a forest of the form as shown in Figure 4.6 (thus y is a member of $\mathcal{G}(u)$ and z is a member of $\mathcal{G}(v)$, with $u \neq v$; u and v are sons of x);*
2. *$\Gamma \vdash \Delta, y : A, z : B$ is not x-branching*

then one of the following sequents is derivable:

(i) $\Gamma^*_u \vdash \Delta^*_u, y : A$

(ii) $\Gamma^*_v \vdash \Delta^*_v, z : B$

(iii) Γ - $(\Gamma^\circ_u \cup \Gamma^\circ_v) \vdash \Delta$ - $(\Delta^\circ_u \cup \Delta^\circ_v)$

*Moreover, the proofs of (i), (ii), and (iii) do not add any application of ($\Rightarrow$ **L**) to the proof of $\Gamma \vdash \Delta, y : A, z : B$.*

Theorem 4.45 (Disjunction property). *Given a non x-branching sequent*

$$\Gamma \vdash \Delta, x : A_1 \Rightarrow B_1, x : A_2 \Rightarrow B_2$$

derivable with a derivation Π, one of the following sequents:

1. $\Gamma \vdash \Delta$*,* $x : A_1 \Rightarrow B_1$
2. $\Gamma \vdash \Delta$*,* $x : A_2 \Rightarrow B_2$

is derivable.

Proof. If $x : A_1 \Rightarrow B_1$ is introduced by weakening, then we obtain a proof of $\Gamma \vdash \Delta, x : A_2 \Rightarrow B_2$ by removing that weakening, and the same for the symmetric case. Otherwise, both conditionals are introduced by ($\Rightarrow$ **R**); by the invertibility of ($\Rightarrow$ **R**), we can consider a proof ended as follows:

$$\cfrac{\cfrac{\Gamma, x \xrightarrow{A_1} y, x \xrightarrow{A_2} z \vdash \Delta, y : B_1, z : B_2}{\Gamma, x \xrightarrow{A_1} y \vdash \Delta, y : B_1, x : A_2 \Rightarrow B_2}\ (\Rightarrow \mathbf{R})}{\Gamma \vdash \Delta, x : A_1 \Rightarrow B_1, x : A_2 \Rightarrow B_2}\ (\Rightarrow \mathbf{R})$$

in which $\Gamma, x \xrightarrow{A_1} y, x \xrightarrow{A_2} z \vdash \Delta, y : B_1, z : B_2$ respects all the conditions to Apply the Proposition 4.44. Therefore, we apply the Proposition 4.44 and in each case we conclude that either $\Gamma \vdash \Delta$, $x : A_1 \Rightarrow B_1$ or $\Gamma \vdash \Delta$, $x : A_2 \Rightarrow B_2$ is derivable. For the entire proof, see Theorem 5.13 in [OPS05].

■

By the soundness and completeness of SeqS, it is easy to prove the following corollary of the disjunction property:

Corollary 4.46. *If $(A \Rightarrow B) \vee (C \Rightarrow D)$ is valid in CK{+MP}{+ID}, then either $A \Rightarrow B$ or $C \Rightarrow D$ is valid in CK{+MP}{+ID}.*

$$(\Rightarrow \mathbf{L})\ \frac{x \xrightarrow{A'} y \vdash x \xrightarrow{A} y \qquad\qquad \Gamma, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta}\ , x \xrightarrow{A'} y \in \Gamma$$

Figure 4.7: ($\Rightarrow$ **L**) rule for SeqCK and SeqID systems.

4.5.1 Refinements for CK{+ID}

In this subsection we give a further refinement for the basic conditional logic CK and its extension CK+ID. The Proposition 4.44 suggests the following fact: in SeqCK and SeqID systems, it is useless to apply ($\Rightarrow$ **L**) on the same formula $x : A \Rightarrow B$ *by using more than one transition* $x \xrightarrow{A} y$, with a different y. Intuitively, if ($\Rightarrow$ **L**) is applied to $x : A \Rightarrow B$ by using two (or more) transitions $x \xrightarrow{A} y$ and $x \xrightarrow{A} z$, then the proof can be directed either on the subtree with root y (i.e. $\mathcal{G}(y)$) or on $\mathcal{G}(z)$. Therefore, to prove $\Gamma, x \xrightarrow{A} y_1, x \xrightarrow{A} y_2, ..., x \xrightarrow{A} y_n, x : A \Rightarrow B \vdash \Delta$ one needs *only one application of* ($\Rightarrow$ **L**) *in each branch.* This means that only one transition $x \xrightarrow{A} y_i$ will be need to apply ($\Rightarrow$ **L**) on $x : A \Rightarrow B$ in each branch[11].

This fact is formalized as follows (the proof can be found in Lemma 5.15 in [OPS05]):

Lemma 4.47 (Controlled used of ($\Rightarrow$ **L**) in SeqCK and SeqID)**.** *If* $\Gamma, x : A \Rightarrow B \vdash \Delta$ *is derivable in SeqCK (SeqID), then it has a derivation with at most one application of (*$\Rightarrow$ **L***) with* $x : A \Rightarrow B$ *as a principal formula in each branch.*

By the above Lemma 4.47 and Theorem 4.19 we can reformulate the ($\Rightarrow$ **L**) rule as shown in Figure 4.7.

As mentioned above, thanks to the reformulation shown in Figure 4.7, it is possible to give a better space complexity bound for CK{+ID}:

Theorem 4.48 (Space complexity of CK{+ID})**.** *Provability in CK{+ID} is decidable in* $O(n\ \log n)$ *space.*

Proof. We proceed exactly in the same way as in the proof of Theorem 4.25. We conclude that provability in CK{+ID} is decidable in $O(n\ \log n)$ space since the length of a branch proof is $O(n)$, therefore to store the whole stack requires $O(n\ \log n)$ space.

■

[11] Observe that the reformulation of the rule would not be complete for the other systems, where we potentially need to apply ($\Rightarrow$ **L**) on $x : A \Rightarrow B$ by using *all* the transitions $x \xrightarrow{A} y_i$ in the left-hand side of the sequent. In systems with (**MP**) $x \xrightarrow{A} x$ can also be used.

4.6 Uniform Proofs

In this section we discuss how our calculi can be used to develop goal-directed proof procedures for conditional logics, following the paradigm of Uniform Proof by Miller and others [MNPS91][12]. A full investigation of this topic could lead to the development of a language for reasoning hypothetically about change and actions in a logic programming framework.

The paradigm of uniform proof is an abstraction, or a generalization, of conventional logic programming. The basic idea of goal-directed proof search is as follows: given a sequent $\Gamma \vdash G$, one can interpret Γ as the program or database, and G as a goal whose proof is searched. The backward proof search of the sequent is driven by the goal G, in the sense that the goal is stepwise decomposed according to its logical structure by the rules of the calculus, until its atomic constituents are reached. To prove an atomic goal Q, the proof search mechanism checks if Γ contains a "clause" whose head matches with Q and then tries to prove the "body" of the clause.

The logical connectives in G can be interpreted operationally as simple and fixed search instructions. In our case, the situation will be as follows: a database consists of formulas labelled by worlds and transitions between worlds labelled by formulas, and we have a goal to be proved in a specific world. In this respect, for instance, the operational meaning of the rule for conditional goals is to expand the database with a new transition. A proof of this sort is called a *uniform proof.* Given a sequent calculus, not every valid sequent admits a uniform proof of this kind; in order to describe a goal-directed proof search one must identify a (significant) fragment of the corresponding logic that allows uniform proofs. Usually, this fragment (in the propositional case) is alike to the Harrop-fragment of intuitionistic logic (see [MNPS91, MH94, PH94]). To specify this fragment one distinguishes between the formulas which can occur in the database (D-formulas) and the formulas that can be asked as goals (G-formulas).

First of all, we specify the fragment of CK{+ID}{+MP} we consider[13]. We distinguish between the formulas which can occur in the program (or knowledge base or database), called D-formulas, and the formulas that can be asked as goals, called G-formulas.

Definition 4.49 (Language for uniform proofs)**.** *We consider the fragment of CK{+ID}{+MP}, called $\mathcal{LU}(CK\{+ID\}\{+MP\})$, comprising:*

- database formulas*, denoted with D*
- goal formulas*, denoted with G*
- transition formulas *of the form $x \xrightarrow{A} y$*

defined as follows ($Q \in ATM$):

$$\begin{aligned}
&D = G \rightarrow Q \mid A \Rightarrow D\\
&G = Q \mid \top \mid G \wedge G \mid G \vee G \mid A \Rightarrow G\\
&A = Q \mid A \wedge A \mid A \vee A
\end{aligned}$$

We define a database Γ *as a set of D-formulas and transition formulas.*

[12] A related methodology for goal-directed provability has been proposed also in [GO00], where goal-directed proof procedures for several families of nonclassical logics are presented.

[13] We present here a fragment of the language that allows uniform proofs and that might have some practical interest. We do not claim that this is a maximal fragment allowing uniform proofs. The determination of a maximal fragment is an issue that we shall explore in future research.

Formulas of kind A can be either atomic formulas or combinations of conjunctions and disjunctions of atomic formulas. It is worth noticing that the fragment considered could be easily extended by allowing formulas of type A of the form $A \Rightarrow A$ and allowing falsity $\bot$ as a head of a D-formula, without changing essentially the proof procedure displayed below. As mentioned above, here we restrict our concern to the basic system CK and its extensions with axioms ID and MP. We denote sequent calculi for these systems with SeqS', i.e. we use SeqS' to refer to all the following systems: SeqCK, SeqID, SeqMP, and SeqID+MP. We call $\mathcal{U}$S' the goal-directed proof procedures introduced for S' systems.

The rules of the calculi $\mathcal{U}$S' are presented in Figure 4.8. $\mathcal{U}$S' 's rules are able to prove a goal which is either a formula $x : G$ or a transition formula $x \xrightarrow{A} y$; from now on we use γ to denote a goal of both kinds. Given a goal γ whose proof is searched from a database Γ, we call $\Gamma \vdash_{GD} \gamma$ a *query* . We write:

$$\Gamma \vdash_{GD} \gamma \Longrightarrow \begin{array}{l} \Gamma_1 \vdash_{GD} \gamma_1 \\ \Gamma_2 \vdash_{GD} \gamma_2 \\ \dots \\ \Gamma_n \vdash_{GD} \gamma_n \end{array}$$

to denote that the sequent $\Gamma \vdash_{GD} \gamma$ is reduced to sequents $\Gamma_i \vdash_{GD} \gamma_i$, with $i = 1, 2, \dots, n$. The rule ($\mathcal{U}$ **prop**) is used when D-formulas have the form $A_1 \Rightarrow A_2 \Rightarrow ... \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, where G could be $\top$.

The rule $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ belongs to $\mathcal{U}$CK and $\mathcal{U}$MP only, whereas $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ is used in $\mathcal{U}$ID and $\mathcal{U}$ID+MP. The rule $(\mathcal{U}$ **trans**$)_{\mathbf{MP}}$ only belongs to the calculi $\mathcal{U}$MP and $\mathcal{U}$ID+MP. The following table shows the rules belonging to each goal-directed calculus:

Calculus	**Rules**
$\mathcal{U}$CK	$(\mathcal{U}\top)$, $(\mathcal{U}$**ax**$)$, $(\mathcal{U}$ **prop**$)$, $(\mathcal{U}\wedge)$, $(\mathcal{U}\vee)$, $(\mathcal{U}$ **trans**$)$, $(\mathcal{U}\Rightarrow)_{\mathbf{CK}}$
$\mathcal{U}$ID	$(\mathcal{U}\top)$, $(\mathcal{U}$**ax**$)$, $(\mathcal{U}$ **prop**$)$, $(\mathcal{U}\wedge)$, $(\mathcal{U}\vee)$, $(\mathcal{U}$ **trans**$)$, $(\mathcal{U}\Rightarrow)_{\mathbf{ID}}$
$\mathcal{U}$MP	$(\mathcal{U}\top)$, $(\mathcal{U}$**ax**$)$, $(\mathcal{U}$ **prop**$)$, $(\mathcal{U}\wedge)$, $(\mathcal{U}\vee)$, $(\mathcal{U}$ **trans**$)$, $(\mathcal{U}\Rightarrow)_{\mathbf{CK}}$, $(\mathcal{U}$ **trans**$)_{\mathbf{MP}}$
$\mathcal{U}$ID+MP	$(\mathcal{U}\top)$, $(\mathcal{U}$**ax**$)$, $(\mathcal{U}$ **prop**$)$, $(\mathcal{U}\wedge)$, $(\mathcal{U}\vee)$, $(\mathcal{U}$ **trans**$)$, $(\mathcal{U}\Rightarrow)_{\mathbf{ID}}$, $(\mathcal{U}$ **trans**$)_{\mathbf{MP}}$

Given a formula of type A, i.e. either an atomic formula or a boolean combination of conjunctions and disjunctions of atomic formulas, the operation $\mathsf{Flat}(x : A)$ has the effect of *flatten* the conjunction/disjunction into a set of atomic D-formulas:

Definition 4.50 (Flat Operation)**.** *We define:*

- $\mathsf{Flat}(x : Q) = \{\{x : Q\}\}$, *with* $Q \in ATM$*;*
- $\mathsf{Flat}(x : F \vee G) = \mathsf{Flat}(F) \cup \mathsf{Flat}(G)$
- $\mathsf{Flat}(x : F \wedge G) = \{S_F \cup S_G \mid S_F \in \mathsf{Flat}(F)$ *and* $S_G \in \mathsf{Flat}(G)\}$

This operation is needed when *rules introducing a formula of type A in the database* are applied, since an A-formula might not be a D-formula. These rules, namely ($\mathcal{U}$ **trans**) and ($\mathcal{U} \Rightarrow)_{\mathbf{ID}}$, introduce a formula of type A in the left-hand side of the sequent, i.e. a formula possibly being a combination of conjunctions and disjunctions of atoms. This formula needs to be decomposed in its atomic components. Indeed, in order to search a derivation for a transition formula $x \xrightarrow{A} y$, when ($\mathcal{U}$ **trans**) is applied by considering a transition $x \xrightarrow{A'} y$ in the program (or database), then the calculus leads to search a derivation for both $u : A' \vdash_{GD} u : A$ and $u : A \vdash_{GD} u : A'$; intuitively, this step corresponds to an application of the (**EQ**) rule in SeqS. As an example, suppose that A has the form $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n$; in this case, in order to prove $u : A \vdash_{GD} u : A'$, we need to flat the database, thus proving the following sequent: $u : Q_1, u : Q_2, \ldots, u : Q_n \vdash_{GD} u : A'$. In case A has the form $Q_1 \vee Q_2$, then we need to prove both $u : Q_1 \vdash u : A'$ and $u : Q_2 \vdash u : A'$. The same happens when $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ is applied to $\Gamma \vdash_{GD} x : A \Rightarrow B$, and the computation steps to prove $\Gamma, x \xrightarrow{A} y, y : A \vdash_{GD} y : B$, and $y : A$ needs to be decomposed as defined here above.

Moreover, we write Γ, $\mathsf{Flat}(x : A) \vdash_{GD} \gamma$ to denote that the goal γ can be derived from *all* the databases obtained by applying Flat to $x : A$ (and adding formulas in Γ), that is to say:

Definition 4.51. *Given a database Γ, a formula A and a goal G, let $\mathsf{Flat}(x : A) = \{S_1, S_2, \ldots, S_n\}$. We write*

$$\Gamma,\ \mathsf{Flat}(x : A) \vdash_{GD} \gamma$$

if and only if

$$\forall i = 1, 2, \ldots, n \textit{ we have that } \Gamma, S_i \vdash_{GD} \gamma$$

We restrict our concern to computations beginning with initial queries. Intuitively, an *initial query* comprises a database of the form $x : D_1, x : D_2, \ldots, x : D_n$, where $D_1, D_2, \ldots, D_n$ are D-formulas, and a goal $x : G$. These queries are the only ones having a direct logical interpretation, namely to prove that G is a logical consequence of $D_1, D_2, \ldots, D_n$.

Definition 4.52 (Initial query)**.** *Given a database $x : D_1, x : D_2, \ldots, x : D_n$, where $D_1, D_2, \ldots, D_n$ are formulas of type D, we call* initial query *a query in which the proof of $x : G$ is searched, where G is a formula of type G.*

We can observe that no rule of the calculi $\mathcal{U}$S' remove a formula from the database. In particular, starting with an initial query, the rules $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ and $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ introduce a transition $x \xrightarrow{A} y$ in the database, where y is a new label. As a direct consequence, the set of transitions in the database forms a *tree*. This is stated by Definitions 4.53 and 4.54 below.

Definition 4.53 (Multigraph of transitions)**.** *Given a database Γ, where $\Gamma = \Gamma', T$ and T is the multiset of transition formulas and Γ' does not contain transition formulas, and a goal γ, i.e. either a formula $x : G$ or a transition $x \xrightarrow{A} y$, we define the multigraph $\mathcal{G} = \langle V, E \rangle$ associated to $\Gamma \vdash_{GD} \gamma$, with vertexes V and edges E. V is the set of labels occurring in $\Gamma \vdash_{GD} \gamma$, and $< u, v > \in E$ whenever $u \xrightarrow{A'} v \in T$.*

We call *regular query* a query $\Gamma \vdash_{GD} \gamma$ whose multiset of transitions forms a tree:

$(\mathcal{U}\top)$	$\Gamma \vdash_{GD} x : \top$		
$(\mathcal{U}$ **ax**$)$	$\Gamma \vdash_{GD} x : Q$, if $x : Q \in \Gamma$		
$(\mathcal{U}$ **prop**$)$	$\Gamma \vdash_{GD} x : Q$ if $y : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q) \in \Gamma$	$\Longrightarrow$	$\Gamma \vdash_{GD} y \xrightarrow{A_1} x_1$ $\Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2$ $\ldots$ $\ldots$ $\Gamma \vdash_{GD} x_{n-1} \xrightarrow{A_n} x$ $\Gamma \vdash_{GD} x : G$
$(\mathcal{U}\wedge)$	$\Gamma \vdash_{GD} x : G_1 \wedge G_2$	$\Longrightarrow$	$\Gamma \vdash_{GD} x : G_1$ $\Gamma \vdash_{GD} x : G_2$
$(\mathcal{U}\vee)$	$\Gamma \vdash_{GD} x : G_1 \vee G_2$	$\Longrightarrow$	$\Gamma \vdash_{GD} x : G_1$
	or		
	$\Gamma \vdash_{GD} x : G_1 \vee G_2$	$\Longrightarrow$	$\Gamma \vdash_{GD} x : G_2$
$(\mathcal{U}$ **trans**$)$	$\Gamma \vdash_{GD} x \xrightarrow{A} y$ if $x \xrightarrow{A'} y \in \Gamma,$ and $x \neq y$	$\Longrightarrow$	$\mathsf{Flat}(u : A') \vdash_{GD} u : A$ $\mathsf{Flat}(u : A) \vdash_{GD} u : A'$
$(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$	$\Gamma \vdash_{GD} x : A \Rightarrow G$ y new	$\Longrightarrow$	$\Gamma, x \xrightarrow{A} y \vdash_{GD} y : G$
$(\mathcal{U}$ **trans**$)_{\mathbf{MP}}$	$\Gamma \vdash_{GD} x \xrightarrow{A} x$	$\Longrightarrow$	$\Gamma \vdash_{GD} x : A$
$(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$	$\Gamma \vdash_{GD} x : A \Rightarrow G$ y new	$\Longrightarrow$	$\Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash_{GD} y : G$

Figure 4.8: The rules for the calculi $\mathcal{U}$S'.

Definition 4.54 (Regular query)**.** *A query* $\Gamma \vdash_{GD} \gamma$ *is called* regular *if the multigraph of transitions* $\mathcal{G}$ *associated with* $\Gamma \vdash_{GD} \gamma$ *is a tree.*

As mentioned, we can show that, given an initial query, the computation leads only to regular queries:

Lemma 4.55. *Every computation started with an initial query and obtained by applying* $\mathcal{U}S$*'s rules contains only regular queries.*

Proof. Given a proof beginning with an initial query, we proceed by induction on the distance between the query we consider and the initial query. For the base case (distance=0), we consider the initial query itself. It only contains one label x, and the database does not contain transitions. Therefore, the multigraph of the transitions is a tree (containing only the vertex x and no edges). For the inductive step, we consider each rule of the calculi $\mathcal{U}$S'. Rules ($\mathcal{U}$ **prop**), ($\mathcal{U}$ $\wedge$), ($\mathcal{U}$ $\vee$), and $(\mathcal{U}\ \textbf{trans})_{\textbf{MP}}$ do not add any formula in the database, therefore the database obtained by an application of these rules is a tree by inductive hypothesis. If $(\mathcal{U}{\Rightarrow})_{\textbf{CK}}$ is applied to $\Gamma \vdash_{GD} x : A \Rightarrow G$, we have a query $\Gamma, x \xrightarrow{A} y \vdash_{GD} y : B$, where y is a new label. By inductive hypothesis, Γ is a tree, and so $\Gamma, x \xrightarrow{A} y$ since x occurs in Γ (no rule removes formulas, thus labels) and y is a new label, then the rule adds both a vertex and an edge to the tree. The same for the rule $(\mathcal{U}{\Rightarrow})_{\textbf{ID}}$. If ($\mathcal{U}$ **trans**) is applied to $\Gamma \vdash x \xrightarrow{A} y$, given $x \xrightarrow{A'} y \in \Gamma$, then the computation leads to prove $\mathsf{Flat}(u : A') \vdash_{GD} u : A$ and $\mathsf{Flat}(u : A) \vdash_{GD} u : A'$: in both cases, the multigraph of transitions only contains a vertex (u) and no edges, i.e. it is a tree.

■

In the following we prove that the calculi $\mathcal{U}$S' are sound and complete with respect to the semantics. These calculi for uniform proofs are grounded on the properties of the corresponding calculi SeqS', when derivations are restricted to the fragment $\mathcal{LU}$(CK{+ID}{+MP}) and to regular sequents (Definition 4.39 in Section 4.5). By Lemma 4.55, in $\mathcal{U}$S' we can restrict our concern to regular queries. Regular queries are obviously regular sequents, then the properties described in Section 4.5 also hold for regular queries. These properties are consequences of the refinements on the calculi SeqS', where the two rules (**ID**) and (**MP**) are reformulated with the non-invertible ones shown in Figure 4.4, that we recall here for a better readability:

$$(\textbf{ID})\ \frac{\Gamma, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y \vdash \Delta} \qquad\qquad (\textbf{MP})\ \frac{\Gamma \vdash x : A, \Delta}{\Gamma \vdash x \xrightarrow{A} x, \Delta}$$

First, we state the following *disjunction property*, whose proof is similar to the one of Theorem 4.45, since that proof is based on the properties of regular sequents, and holding also for systems with axioms ID and MP: if $\Gamma \vdash x : A, y : B$ is derivable, then either $\Gamma \vdash x : A$ or $\Gamma \vdash y : B$ is derivable. Moreover, this property is *height-preserving*, in the sense that the derivation of $\Gamma \vdash x : A$ (respectively $\Gamma \vdash y : B$) has no greater height than $\Gamma \vdash x : A, y : B$. Notice that $x : A$ and $y : B$ are *not* necessarily conditional formulas. This property is stated as follows:

Proposition 4.56 (Height-preserving disjunction property)**.** *Let Γ be a database. If $\Gamma \vdash x_1 : G_1, x_2 : G_2, ..., x_n : G_n$ is derivable in SeqS' with a proof of height h, then there exists $i, i = 1, 2, ..., n$, such that $\Gamma \vdash x_i : G_i$ is derivable in SeqS' with a proof of no greater height than h.*

Now we have all the elements to prove that $\mathcal{U}$S' is sound and complete with respect to the semantics.

Theorem 4.57 (Soundness of $\mathcal{U}$S')**.** *If Γ is a database, γ is a goal (i.e. either a formula $x : G$ or a transition formula $x \xrightarrow{A} y$), and $\Gamma \vdash_{GD} \gamma$ is derivable in $\mathcal{U}$S', then $\Gamma \vdash \gamma$ is derivable in the corresponding system SeqS'.*

Proof. By induction on the height of the derivation of $\Gamma \vdash_{GD} \gamma$ in $\mathcal{U}$S'.

In the base case, we have that either (1) the goal G is $\top$, then $\Gamma \vdash x : \top$ is also derivable in SeqS', or (2) we have that $\Gamma \vdash_{GD} x : Q$ and $x : Q \in \Gamma$, then $\Gamma \vdash x : Q$ is also derivable in SeqS'. Indeed, $\Gamma \vdash x : \top$ (respectively $\Gamma \vdash x : Q$ with $x : Q \in \Gamma$) is an instance of (AX). For the inductive step, we have to consider each rule of $\mathcal{U}$S' applied to $\Gamma \vdash_{GD} x : G$. To save space, we only present the most interesting cases of ($\mathcal{U}$ **prop**), $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$, and $(\mathcal{U}\ \mathbf{trans})_{\mathbf{MP}}$. The other cases are similar and left to the reader.

- the proof of $\Gamma \vdash_{GD} x : Q$ is ended by an application of ($\mathcal{U}$ **prop**), i.e. $\Gamma \vdash_{GD} y \xrightarrow{A_1} x_1$, $\Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2, \ldots, \Gamma \vdash_{GD} x_{n-1} \xrightarrow{A_n} x$ are derivable in $\mathcal{U}$S', and so $\Gamma \vdash_{GD} x : G$. Moreover, we have that $y : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \to Q) \in \Gamma$. By inductive hypothesis, we have that $(1)\Gamma \vdash y \xrightarrow{A_1} x_1$, $(2)\Gamma \vdash x_1 \xrightarrow{A_2} x_2, \ldots, (n)\Gamma \vdash x_{n-1} \xrightarrow{A_n} x$, and $(*)\Gamma \vdash x : G$ are derivable in SeqS'. By $(*)$ and the fact that weakening is admissible in SeqS' (Theorem 4.10), we have that $(**)\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \to Q), \ldots, x_n : A_n \Rightarrow (G \to Q) \vdash x : G, x : Q$ is also derivable in SeqS'. We have also that $(***)\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \to Q), \ldots, x_n : A_n \Rightarrow (G \to Q), x : Q \vdash x : Q$ has a derivation in SeqS', since it is an instance of (AX). We can conclude as follows:

$$\dfrac{\dfrac{(**)\Gamma\ldots \vdash x : G, x : Q \qquad (***)\Gamma\ldots x : Q \vdash x : Q}{(n')\Gamma\ldots \vdash x_{n-1} \xrightarrow{A_n} x, x : Q \mid \Gamma\ldots x : G \to Q \vdash x : Q}\ (\to L)}{\Gamma\ldots x_{n-1} : A_n \Rightarrow (G \to Q) \vdash x : Q}\ (\Rightarrow L)$$

$$\dfrac{(1')\Gamma \vdash y \xrightarrow{A_1} x_1, x : Q \qquad \dfrac{(2')\Gamma\ldots \vdash x_1 \xrightarrow{A_2} x_2, x : Q \qquad \begin{array}{c}\vdots\\ \Gamma\ldots x_2 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \to Q) \vdash x : Q\end{array}}{\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \to Q) \vdash x : Q}\ (\Rightarrow \mathbf{L})}{\Gamma \vdash x : Q}\ (\Rightarrow \mathbf{L})$$

  where $(1'), (2'), \ldots, (n')$ can be obtained from $(1), (2), \ldots, (n)$ since weakening is height preserving admissible in SeqS';

- the proof of $\Gamma \vdash_{GD} x : A \Rightarrow G$ is ended by an application of $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$, i.e. $\Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash_{GD} y : G$ has a derivation in $\mathcal{U}$ID{+MP}. By inductive hypothesis, we have that $\Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash y : G$ is derivable in SeqID{+MP}, from we can conclude that $\Gamma, x \xrightarrow{A} y, y : A \vdash y : G$ is derivable in SeqS' by applying the rules ($\wedge$ **L**) and ($\vee$ **L**). As an example, suppose that A is a formula of the kind $A_1 \wedge (A_2 \vee A_3)$, then $\mathsf{Flat}(y : A) = \{\{y : A_1, y :$

$A_2\}, \{y : A_1, y : A_3\}\}$. By inductive hypothesis we have that the sequents $\Gamma, x \xrightarrow{A} y, y : A_1, y : A_2 \vdash y : G$ and $\Gamma, x \xrightarrow{A} y, y : A_1, y : A_3 \vdash y : G$ are both derivable in SeqS'; by applying ($\wedge$ **L**) and ($\vee$ **L**) we can find the following derivation of $\Gamma, x \xrightarrow{A} y, y : A \vdash y : G$ in SeqS':

$$\dfrac{\dfrac{\Gamma, x \xrightarrow{A} y, y : A_1, y : A_2 \vdash y : G \qquad \Gamma, x \xrightarrow{A} y, y : A_1, y : A_3 \vdash y : G}{\Gamma, x \xrightarrow{A} y, y : A_1, y : A_2 \vee A_3 \vdash y : G}\ (\vee\mathbf{L})}{\Gamma, x \xrightarrow{A} y, y : A_1 \wedge (A_2 \vee A_3) \vdash y : G}\ (\wedge\mathbf{L})$$

Since $\Gamma, x \xrightarrow{A} y, y : A \vdash y : G$ is derivable in SeqS, we can conclude by an application of (**ID**), followed by an application of ($\Rightarrow$ **R**), since by definition of $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ the label y is new:

$$\begin{array}{cl} \Gamma, x \xrightarrow{A} y, y : A \vdash y : G & \\ & (\mathbf{ID}) \\ \Gamma, x \xrightarrow{A} y \vdash y : G & \\ & (\Rightarrow \mathbf{R}) \\ \Gamma \vdash x : A \Rightarrow G & \end{array}$$

- the proof of $\Gamma \vdash_{GD} x \xrightarrow{A} x$ is ended by an application of $(\mathcal{U}\ \mathbf{trans})_{\mathbf{MP}}$, i.e. $\Gamma \vdash_{GD} x : A$ has a derivation in $\mathcal{U}$\{ID+\}MP. By inductive hypothesis, $\Gamma \vdash x : A$ is derivable in Seq\{ID+\}MP, and so $\Gamma \vdash x \xrightarrow{A} x, x : A$ since weakening is admissible. We can therefore conclude that there is a closed tree for $\Gamma \vdash x \xrightarrow{A} x$ by applying (**MP**) as follows:

$$\begin{array}{cl} \Gamma \vdash x \xrightarrow{A} x, x : A & \\ & (\mathbf{MP}) \\ \Gamma \vdash x \xrightarrow{A} x & \end{array}$$

■

The soundness with respect to the semantics immediately follows from the fact that SeqS' calculi are sound with respect to selection function models.

In order to prove the completeness of $\mathcal{U}$S' we also need to prove the following lemma:

Lemma 4.58. *Given a goal γ and a database Γ, if the following conditions hold:*

- $x_0 : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q) \in \Gamma$*;*
- $(*)$ $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$*,* $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots,$ $x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} \gamma$ *is derivable in $\mathcal{U}$S';*
- $\Gamma \vdash_{GD} x_0 \xrightarrow{A_1} x_1, \Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2, \ldots \Gamma \vdash_{GD} x_{n-1} \xrightarrow{A_n} x_n$ *are derivable in $\mathcal{U}$S'*

then $\Gamma \vdash_{GD} \gamma$ is also derivable in $\mathcal{U}$S'.

Proof. By induction on the height of the uniform proof of $(*)$. In the base case, we have that $(*)$ is an instance of ($\mathcal{U}$ **ax**), that is to say:

- the goal γ has the form $x : P$, with $P \in ATM$;
- there exists $x : P \in \Gamma$.

It is easy to conclude that also $\Gamma \vdash_{GD} x : P$ is an instance of ($\mathcal{U}$ **ax**), then we are done.
Otherwise, $\gamma = x : \top$, and we obviously conclude that $\Gamma \vdash_{GD} x : \top$ is also derivable.

For the inductive step, we consider each rule of $\mathcal{U}$S' that can be applied as the last step of the derivation of $(*)$.

- ($\mathcal{U}\ \wedge$) applied to $\gamma = x : G_1 \wedge G_2$: we have that $(*)$ is obtained by an application of ($\mathcal{U}\ \wedge$) from (1) $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G_1$ and (2) $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G_2$, respectively. Since the proofs of (1) and (2) have a smaller height than $(*)$, we can apply the inductive hypothesis to observe that $\Gamma \vdash_{GD} x : G_1$ and $\Gamma \vdash_{GD} x : G_2$ are derivable in $\mathcal{U}$S'. We can conclude that $\Gamma \vdash_{GD} x : G_1 \wedge G_2$ is also derivable by an application of ($\mathcal{U}\ \wedge$);
- ($\mathcal{U}\ \vee$) applied to $\gamma = x : G_1 \vee G_2$: we have that either (1) $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G_1$ or (2) $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G_2$ are derivable in $\mathcal{U}$S' (and with proofs with smaller height than $(*)$). Suppose that (1) is derivable in $\mathcal{U}$S' (the case in which (2) is derivable is symmetric): we can apply the inductive hypothesis, to obtain that $\Gamma \vdash_{GD} x : G_1$ is derivable in $\mathcal{U}$S', then we can conclude that $\Gamma \vdash_{GD} x : G_1 \vee G_2$ is derivable by an application of ($\mathcal{U}\ \vee$);
- ($\mathcal{U}$ **trans**) applied to $\gamma = x \xrightarrow{A} y$, with $x \neq y$: in this case, we have that there is $x \xrightarrow{A'} y \in \Gamma$ and that $\mathsf{Flat}(u : A') \vdash_{GD} u : A$ and $\mathsf{Flat}(u : A) \vdash_{GD} u : A'$ are derivable in $\mathcal{U}$S'. It is easy to observe that the conditional formulas $x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots,$ $x_{n-1} : A_n \Rightarrow (G \rightarrow Q)$ are not used in an application of ($\mathcal{U}$ **trans**). Therefore, we can conclude that $\Gamma \vdash_{GD} x \xrightarrow{A} y$ is derivable in $\mathcal{U}$S' by applying ($\mathcal{U}$ **trans**) to $\mathsf{Flat}(u : A') \vdash_{GD} u : A$ and $\mathsf{Flat}(u : A) \vdash_{GD} u : A'$;
- ($\mathcal{U}$ **trans**)$_{\mathbf{MP}}$ applied to $\gamma = x \xrightarrow{A} x$: we have that $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : A$ is derivable with a derivation of a smaller height than $(*)$, then we can apply the inductive hypothesis. We have that $\Gamma \vdash_{GD} x \xrightarrow{A} x$ is derivable in $\mathcal{U}$S', and we conclude by an application of ($\mathcal{U}$ **trans**)$_{\mathbf{MP}}$;
- ($\mathcal{U} \Rightarrow$)$_{\mathbf{CK}}$ applied to $\gamma = x : A \Rightarrow G$: we have that $(*)$ $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : A \Rightarrow G$ has been derived from $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q), x \xrightarrow{A} y \vdash_{GD} y : G$, where y does not occur in Γ and it is different from $x_0, x_1, \ldots, x_n$. We can then apply the inductive hypothesis, obtaining that $\Gamma, x \xrightarrow{A} y \vdash_{GD} y : G$ is derivable, and we are done by an application of ($\mathcal{U} \Rightarrow$)$_{\mathbf{CK}}$;
- ($\mathcal{U} \Rightarrow$)$_{\mathbf{ID}}$ applied to $\gamma = x \xrightarrow{A} y$: the proof is as for the case of ($\mathcal{U} \Rightarrow$)$_{\mathbf{CK}}$, with the only difference that we have that, by inductive hypothesis, $\Gamma, \mathsf{Flat}(y : A), x \xrightarrow{A} y \vdash_{GD} y : G$ is derivable in $\mathcal{U}$S';
- ($\mathcal{U}$ **prop**) is applied to $\gamma = x : P$, with $P \in ATM$: we have to distinguish two different cases:

- P is different from Q: in this case, the rule ($\mathcal{U}$ **prop**) ending the derivation of $(*)$ has been applied by using a clause $y : B_1 \Rightarrow B_2 \Rightarrow \ldots \Rightarrow B_m \Rightarrow (G' \rightarrow P) \in \Gamma$, and we have that $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} y \xrightarrow{B_1} y_1$, $\ldots$, $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} y_{m-1} \xrightarrow{B_m} x$, and $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G'$ are derivable in $\mathcal{U}$S'. By inductive hypothesis, we have that $\Gamma \vdash_{GD} y \xrightarrow{B_1} y_1$, $\ldots$, $\Gamma \vdash_{GD} y_{m-1} \xrightarrow{B_m} x$, and $\Gamma \vdash_{GD} x : G'$ are derivable in $\mathcal{U}$S', then we conclude that also $\Gamma \vdash_{GD} x : P$ is derivable by applying the ($\mathcal{U}$ **prop**) rule;
- ($\mathcal{U}$ **prop**) is applied to $\gamma = x : Q$: here we consider two further alternatives. In the first one, the ($\mathcal{U}$ **prop**) rule is applied by using a clause of the database different from all the clauses $x_i : A_{i+1} \Rightarrow \ldots \Rightarrow (G \rightarrow Q)$, with $i = 0, 1, \ldots, n-1$: in this case, we conclude as in the case when ($\mathcal{U}$ **prop**) is applied to $\gamma = x : P$, with P different from Q.

  In the second case, we have that a clause $x_i : A_{i+1} \Rightarrow \ldots \Rightarrow (G \rightarrow Q)$, $i = 0, 1, \ldots, n-1$ is involved in the application of ($\mathcal{U}$ **prop**), then we proceed as follows. First, we observe that $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x_i \xrightarrow{A_{i+1}} x'$; $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x' \xrightarrow{A_{i+2}} x''$; $\ldots$; $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x^\circ \xrightarrow{A_n} x$ are derivable with proofs of a smaller height than $(*)$. We can the apply the inductive hypothesis, and we obtain that also $\Gamma \vdash_{GD} x_i \xrightarrow{A_{i+1}} x'$; $\Gamma \vdash_{GD} x' \xrightarrow{A_{i+2}} x''$; $\ldots$; $\Gamma \vdash_{GD} x^\circ \xrightarrow{A_n} x$ are derivable in $\mathcal{U}$S'. Moreover, we have that $\Gamma, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, $x_2 : A_3 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q), \ldots, x_{n-1} : A_n \Rightarrow (G \rightarrow Q) \vdash_{GD} x : G$ is derivable with no greater height than $(*)$, then we apply the inductive hypothesis to obtain that $(**)$ $\Gamma \vdash_{GD} x : G$ is derivable in $\mathcal{U}$S'.

  By the initial hypothesis, we have that $\Gamma \vdash_{GD} x_0 \xrightarrow{A_1} x_1$, $\Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2$, $\ldots$, $\Gamma \vdash_{GD} x_{i-1} \xrightarrow{A_i} x_i$ are also derivable in $\mathcal{U}$S'. We conclude by an application of ($\mathcal{U}$ **prop**) to $\Gamma \vdash_{GD} x_0 \xrightarrow{A_1} x_1$, $\Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2$, $\ldots$, $\Gamma \vdash_{GD} x_{i-1} \xrightarrow{A_i} x_i$, $\Gamma \vdash_{GD} x_i \xrightarrow{A_{i+1}} x'$, $\Gamma \vdash_{GD} x' \xrightarrow{A_{i+2}} x''$, $\ldots$, $\Gamma \vdash_{GD} x^\circ \xrightarrow{A_n} x$, and $(**)$ $\Gamma \vdash_{GD} x : G$, obtaining that $\Gamma \vdash_{GD} x : Q$ is derivable in $\mathcal{U}$S'.

■

Now we can prove the completeness of the goal-directed calculi.

Theorem 4.59 (Completeness of $\mathcal{U}$S')**.** *If* Γ *is a database,* γ *is a goal (i.e. either a formula* $x : G$ *or a transition* $x \xrightarrow{A} y$*), and* $\Gamma \vdash \gamma$ *is derivable in SeqS', then* $\Gamma \vdash_{GD} \gamma$ *is derivable in* $\mathcal{U}$*S'.*

Proof. By induction on the height of the proof tree in SeqS' for $\Gamma \vdash x : G$. For the base case, we have that if $G = \top$, then we are done by the rule ($\mathcal{U}$ $\top$), otherwise if G

is an atom P, then $x : G$ must belong to Γ and we are done by applying the rule ($\mathcal{U}$ **ax**). For the inductive step, we consider all the cases:

- $\Gamma \vdash x : G_1 \wedge G_2$: since the rule ($\wedge$ **R**) is height-preserving invertible (Theorem 4.11 can be easily extended to the rules for boolean connectives) in SeqS', we can consider a proof ended as follows:

$$\dfrac{\Gamma \vdash x : G_1 \qquad \Gamma \vdash x : G_2}{\Gamma \vdash x : G_1 \wedge G_2}\ (\wedge\mathbf{R})$$

  By inductive hypothesis, $\Gamma \vdash_{GD} x : G_1$ and $\Gamma \vdash_{GD} x : G_2$ are derivable in $\mathcal{U}$S', then we conclude by an application of ($\mathcal{U}$ $\wedge$);

- $\Gamma \vdash x : G_1 \vee G_2$: since ($\vee$ **R**) is height-preserving invertible, there exists a proof ended with an application of ($\vee$ **R**) to $x : G_1 \vee G_2$:

$$\dfrac{\Gamma \vdash x : G_1, x : G_2}{\Gamma \vdash x : G_1 \vee G_2}\ (\vee\mathbf{R})$$

  By Proposition 4.56, we can observe that either $\Gamma \vdash x : G_1$ or $\Gamma \vdash x : G_2$ is derivable with a proof of (at most) the same height, thus we can apply the inductive hypothesis obtaining a derivation in $\mathcal{U}$S' of either $\Gamma \vdash_{GD} x : G_1$ or $\Gamma \vdash_{GD} x : G_2$, then we can conclude by an application of ($\mathcal{U}$ $\vee$);

- $\Gamma \vdash x : A \Rightarrow G$: we have to distinguish the cases of (i) $\mathcal{U}$CK and $\mathcal{U}$MP, comprising the rule $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$, and (ii) $\mathcal{U}$ID and $\mathcal{U}$ID+MP, comprising the rule $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$.

  (i) the rule ($\Rightarrow$ **R**) is height-preserving invertible in SeqCK (resp. SeqMP), therefore we can assume, without loss of generality, that there is a derivation of $\Gamma \vdash x : A \Rightarrow G$ ending as follows:

$$\dfrac{\Gamma, x \xrightarrow{A} y \vdash y : G}{\Gamma \vdash x : A \Rightarrow G}\ (\Rightarrow\mathbf{R})$$

  By inductive hypothesis, $\Gamma, x \xrightarrow{A} y \vdash_{GD} y : G$ is also derivable in $\mathcal{U}$CK (resp. $\mathcal{U}$MP), then we can conclude by an application of $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ since y is a new label in an application of ($\Rightarrow$ **R**);

  (ii) since both the rules ($\Rightarrow$ **R**) and (**ID**) are height-preserving invertible in SeqS', we can consider a proof tree ended with an application of (**ID**), immediately followed by an application of ($\Rightarrow$ **R**), as follows:

$$\dfrac{\dfrac{\Gamma, x \xrightarrow{A} y, y : A \vdash y : G}{\Gamma, x \xrightarrow{A} y \vdash y : G}\ (\mathbf{ID})}{\Gamma \vdash x : A \Rightarrow G}\ (\Rightarrow\mathbf{R})$$

  If A is a boolean combination of disjunctions/conjunctions, then there is a proof tree of $\Gamma, x \xrightarrow{A} y, y : A \vdash y : G$ in SeqS' ended by several applications of ($\vee$ **L**) and ($\wedge$ **L**), since these rules are height-preserving invertible in SeqS' (Theorem 4.11), according to the connectives in A. As an example, let A be $A_1 \vee A_2$: we have that $\Gamma, x \xrightarrow{A_1 \vee A_2} y, y : A_1 \vdash y : G$

and $\Gamma, x \xrightarrow{A_1 \vee A_2} y, y : A_2 \vdash y : G$ are derivable in SeqS'. By inductive hypothesis, we have that all the sequents obtained by applying $(\vee\ \mathbf{L})$ and $(\wedge\ \mathbf{L})$ to $y : A$ (looking backward) are also derivable in $\mathcal{U}$ID{+MP}; in the above example, $\Gamma, x \xrightarrow{A_1 \vee A_2} y, y : A_1 \vdash_{GD} y : G$ and $\Gamma, x \xrightarrow{A_1 \vee A_2} y, y : A_2 \vdash_{GD} y : G$ are derivable in $\mathcal{U}$S', and this corresponds to the fact that $\Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash_{GD} y : G$ is derivable in $\mathcal{U}$S'. We then conclude by an application of $(\mathcal{U}{\Rightarrow})_{\mathbf{ID}}$;

- $\Gamma \vdash x : Q, Q \in ATM$: by Proposition 4.40, we can restrict our concern to regular sequents, then $\Gamma \vdash x : Q$ is a sequent whose multigraph of transitions is a forest. Let Γ^*_x be the multiset of formulas in Γ whose labels occur either on the path from the root of the tree containing x and the label x itself, or in the tree having x as a root. It can be shown that $\Gamma^*_x \vdash x : Q$ is also derivable in SeqS', and with a proof of no greater height than $\Gamma \vdash x : Q$'s. One can observe the following fact: if $\Gamma^*_x \vdash x : Q$ is derivable in SeqS', then there exists $x_0 : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_k \Rightarrow (G^{'} \rightarrow Q) \in \Gamma^*_x$ such that $\Gamma^*_x \vdash x_0 \xrightarrow{A_1} x_1$ and $\Gamma^*_x \vdash x_1 \xrightarrow{A_2} x_2$ and $\ldots$ and $\Gamma^*_x \vdash x_{k-1} \xrightarrow{A_k} x_k$, where $x_k = x$, are also derivable in SeqS' with proofs of no greater height than $\Gamma^*_x \vdash x : Q$. By this fact, we conclude that there is a proof of $\Gamma^*_x \vdash x : Q$ ended by several applications of $(\Rightarrow\ \mathbf{L})$ as follows (we denote with Δ the set of conditionals $x_1 : A_2 \Rightarrow A_3 \Rightarrow \ldots \Rightarrow A_k \Rightarrow (G' \rightarrow Q), x_2 : A_3 \Rightarrow \ldots \Rightarrow A_k \Rightarrow (G' \rightarrow Q), \ldots, x_{k-1} : A_k \Rightarrow (G' \rightarrow Q)$):

$$\dfrac{(1)\Gamma^*_x \vdash x_0 \xrightarrow{A_1} x_1, x : Q \qquad \dfrac{(2)\Gamma^*_x, x_1 : \ldots \vdash x_1 \xrightarrow{A_2} x_2, x : Q \qquad \begin{array}{c} (k)\Gamma^*_x, \Delta \vdash x_{k-1} \xrightarrow{A_k} x, x : Q \\ (\circ)\Gamma^*_x, \Delta, x : G^{'} \rightarrow Q \vdash x : Q \quad (\Rightarrow L) \\ \vdots \\ \Gamma^*_x, x_1 : A_2 \Rightarrow \ldots, x_2 : A_3 \Rightarrow \ldots \vdash x : Q \end{array}}{\Gamma^*_x, x_1 : A_2 \Rightarrow \ldots \Rightarrow A_k \Rightarrow (G^{'} \rightarrow Q) \vdash x : Q}\ (\Rightarrow L)}{\Gamma^*_x \vdash x : Q}\ (\Rightarrow \mathbf{L})$$

where $(1), (2), \ldots (k)$ are derivable in SeqS' since they can be obtained by weakening from $\Gamma^*_x \vdash x_0 \xrightarrow{A_1} x_1$, $\Gamma^*_x \vdash x_1 \xrightarrow{A_2} x_2$, $\ldots$, $\Gamma^*_x \vdash x_{k-1} \xrightarrow{A_k} x_k$, respectively. Again by weakening, we can also obtain that $\Gamma \vdash x_0 \xrightarrow{A_1} x_1$, $\Gamma \vdash x_1 \xrightarrow{A_2} x_2$, $\ldots$, $\Gamma \vdash x_{k-1} \xrightarrow{A_k} x$ are also derivable in SeqS', and with derivations of no greater height than $\Gamma \vdash x : Q$. By inductive hypothesis, we can therefore conclude that $(1')\Gamma \vdash_{GD} x_0 \xrightarrow{A_1} x_1$, $(2')\Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2$, $\ldots$, $(k')\Gamma \vdash_{GD} x_{k-1} \xrightarrow{A_k} x$ are also derivable in $\mathcal{U}$S'.

Since $(\rightarrow \mathbf{L})$ is height-preserving invertible in SeqS', from $(\circ)$ we can find a proof with at most the same height of $\Gamma^*_x, \Delta \vdash x : G^{'}, x : Q$, then of $(\circ\circ)\Gamma, \Delta \vdash x : G^{'}, x : Q$ since weakening is height preserving admissible in SeqS'. By applying the height-preserving disjunction property (Proposition 4.56) to $(\circ\circ)$ we can find a proof of either $\Gamma, \Delta \vdash x : Q$ or $\Gamma, \Delta \vdash x : G^{'}$, to which we can apply the inductive hypothesis obtaining a proof in $\mathcal{U}$S' of either $(i)\Gamma, \Delta \vdash_{GD} x : Q$ or $(ii)\Gamma, \Delta \vdash_{GD} x : G^{'}$, respectively. Moreover, since $(1')$, $(2')$, $\ldots$, and (k') are derivable in $\mathcal{U}$S', by Lemma 4.58 we have that either $(i')\Gamma \vdash_{GD} x : Q$ is derivable in $\mathcal{U}$S' or $(ii')\Gamma \vdash_{GD} x : G'$ is derivable in $\mathcal{U}$S'. In case (i') we are done. In case (ii'), since we have proofs of $(1')\Gamma \vdash x_0 \xrightarrow{A_1} x_1, ..., (k')\Gamma \vdash x_{k-1} \xrightarrow{A_k} x$, we can conclude by an application of ($\mathcal{U}$ **prop**) on these ones and on (ii').

- $\Gamma \vdash x \xrightarrow{A} y$, with $x \neq y$: in Section 4.5, we have proved that a transition formula $x \xrightarrow{A} y$, with $x \neq y$, in the right-hand side of a sequent can only be proved by an application of (**EQ**) as follows:

  $$\dfrac{u : A \vdash u : A' \qquad u : A' \vdash u : A}{\Gamma \vdash x \xrightarrow{A} y}\ (\mathbf{EQ})$$

  if there is $x \xrightarrow{A'} y \in \Gamma$. In the fragment of the language we are considering, transition formulas have the form $x \xrightarrow{A} y$, where A is either an atomic formula or a combination of conjunctions and disjunctions of atomic formulas. Since the rules ($\wedge$ **L**) and ($\vee$ **L**) are both height-preserving invertible in SeqS', we can consider a proof of at most the same height of $u : A \vdash u : A'$ ended (looking forward) by several applications of ($\wedge$ **L**) and ($\vee$ **L**), according to each formula in A. The same for the proof of $u : A' \vdash u : A$.

  As an example, let A be $Q_1 \wedge (Q_2 \vee Q_3)$, and let A' be $P_1 \vee ((P_2 \vee P_3) \wedge P_4)$. Let $\Gamma = \Gamma', x \xrightarrow{A'} y$. We can consider a proof tree ending as follows:

  $$\dfrac{\begin{array}{cc} \begin{array}{c} u : P_2, u : P_4 \vdash u : A \\ u : P_3, u : P_4 \vdash u : A \\ \hline u : P_2 \vee P_3, u : P_4 \vdash u : A \ (\vee\mathbf{L}) \\ \hline u : P_1 \vdash u : A \quad u : (P_2 \vee P_3) \wedge P_4 \vdash u : A \ (\wedge\mathbf{L}) \\ \hline u : P_1 \vee ((P_2 \vee P_3) \wedge P_4) \vdash u : A \ (\vee\mathbf{L}) \end{array} & \begin{array}{c} u : Q_1, u : Q_2 \vdash u : A' \\ u : Q_1, u : Q_3 \vdash u : A' \\ \hline u : Q_1, u : Q_2 \vee Q_3 \vdash u : A' \ (\vee\mathbf{L}) \\ \hline u : Q_1 \wedge (Q_2 \vee Q_3) \vdash u : A' \ (\wedge\mathbf{L}) \end{array} \end{array}}{\Gamma', x \xrightarrow{A'} y \vdash x \xrightarrow{A} y}\ (\mathbf{EQ})$$

  By inductive hypothesis, all sequents whose antecedents have been decomposed in their atomic components are also derivable in $\mathcal{U}$S'. In the above example, we have that $u : P_1 \vdash_{GD} u : A; u : P_2, u : P_4 \vdash_{GD} u : A; u : P_3, u : P_4 \vdash_{GD} u : A; u : Q_1, u : Q_2 \vdash_{GD} u : A'$; and $u : Q_1, u : Q_3 \vdash_{GD} u : A'$ are derivable in $\mathcal{U}$S'. All the databases of these sequents correspond to the elements of the sets obtained by applying Flat to $u : A$ and $u : A'$, that is to say $\mathsf{Flat}(u : A) \vdash_{GD} u : A'$ and $\mathsf{Flat}(u : A') \vdash_{GD} u : A$ are derivable, then we can conclude by an application of ($\mathcal{U}$ **trans**).

- $\Gamma \vdash x \xrightarrow{A} x$: by the reformulation of (**MP**) shown in Figure 4.4, we can assume, without loss of generality, that there is a derivation in SeqS' ending as follows:

  $$\dfrac{\Gamma \vdash x : A}{\Gamma \vdash x \xrightarrow{A} x}\ (\mathbf{MP})$$

  By inductive hypothesis, we have a derivation in $\mathcal{U}$S' of $\Gamma \vdash_{GD} x : A$, then we can conclude that also $\Gamma \vdash_{GD} x \xrightarrow{A} x$ is derivable in $\mathcal{U}$S' by an application of ($\mathcal{U}$ **trans**)$_{\mathbf{MP}}$.

■

The completeness with respect to the semantics immediately follows from the fact that SeqS' calculi are complete with respect to selection function models.

4.6.1 Examples in $\mathcal{U}$S'

As a first example of usage of the goal-directed proof procedures $\mathcal{U}$S', consider the following knowledge base Γ, representing the functioning of a simple DVD recorder:

- (1) $x : seton \Rightarrow ((pressRecButton \Rightarrow recording) \rightarrow readyToRec)$
- (2) $x : seton \Rightarrow pressRecButton \Rightarrow (sourceSelected \rightarrow recording)$
- (3) $x : seton \Rightarrow pressRecButton \Rightarrow (\top \rightarrow sourceSelected)$

We should interpret a conditional formula $A \Rightarrow B$ as "if the current state is updated with A then B holds" or, if A were an action, as "as an effect of A, B holds", or "having performed A, B holds". In this respect, clauses in Γ can be interpreted as: (1) "having set the device on, it is ready to record whenever it starts to record after pressing the REC button"; (2) "having set the device on and then having pressed the REC button, if the registration source has been selected, then the device will start to record"; (3) "having set the device on and then having pressed the REC button, it results that the registration source has been selected (the default source)". For a broader discussion on plausible interpretations of conditionals, we remind the reader to [Sch99, GS04, GGM$^+$00]; here we just observe that they have been widely used to express update/action/causation.

We show that the goal "Having set the DVD recorder on, is it ready to record?", formalized as follows:

$$x : seton \Rightarrow readyToRec$$

derives from Γ in $\mathcal{U}$CK. The derivation is presented in Figure 4.9.
As another example, consider the following database Γ:

- (1) $x : A \Rightarrow (B \vee C) \Rightarrow (D \rightarrow E)$
- (2) $x : A \Rightarrow (((B \vee C) \Rightarrow E) \rightarrow F)$
- (3) $x : A \Rightarrow (B \vee C) \Rightarrow (B \vee G) \Rightarrow (\top \rightarrow D)$
- (4) $x : A \Rightarrow (B \vee C) \Rightarrow (C \vee H) \Rightarrow (\top \rightarrow D)$

In $\mathcal{U}$ID+MP one can show that the goal

$$x : A \Rightarrow F$$

can be derived by the above database. A derivation is presented in Figure 4.10.

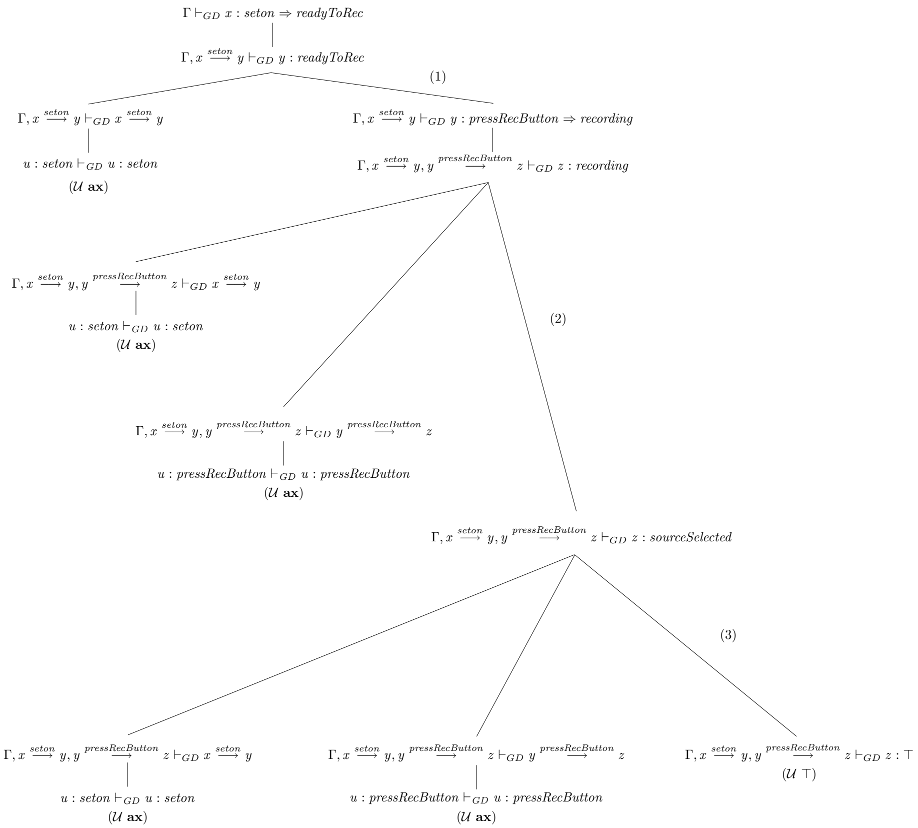

$\Gamma \vdash_{GD} x : seton \Rightarrow readyToRec$
$\Gamma, x \xrightarrow{seton} y \vdash_{GD} y : readyToRec$
(1)
$\Gamma, x \xrightarrow{seton} y \vdash_{GD} x \xrightarrow{seton} y$
$u : seton \vdash_{GD} u : seton$
$(\mathcal{U}\ \mathbf{ax})$
$\Gamma, x \xrightarrow{seton} y \vdash_{GD} y : pressRecButton \Rightarrow recording$
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} z : recording$
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} x \xrightarrow{seton} y$
$u : seton \vdash_{GD} u : seton$
$(\mathcal{U}\ \mathbf{ax})$
(2)
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} y \xrightarrow{pressRecButton} z$
$u : pressRecButton \vdash_{GD} u : pressRecButton$
$(\mathcal{U}\ \mathbf{ax})$
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} z : sourceSelected$
(3)
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} x \xrightarrow{seton} y$
$u : seton \vdash_{GD} u : seton$
$(\mathcal{U}\ \mathbf{ax})$
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} y \xrightarrow{pressRecButton} z$
$u : pressRecButton \vdash_{GD} u : pressRecButton$
$(\mathcal{U}\ \mathbf{ax})$
$\Gamma, x \xrightarrow{seton} y, y \xrightarrow{pressRecButton} z \vdash_{GD} z : \top$
$(\mathcal{U}\ \top)$

Figure 4.9: A derivation of the goal $x : seton \Rightarrow readyToRec$. When **($\mathcal{U}$ prop)** is applied, the corresponding edge is labeled with the number of the formula of the initial database used.

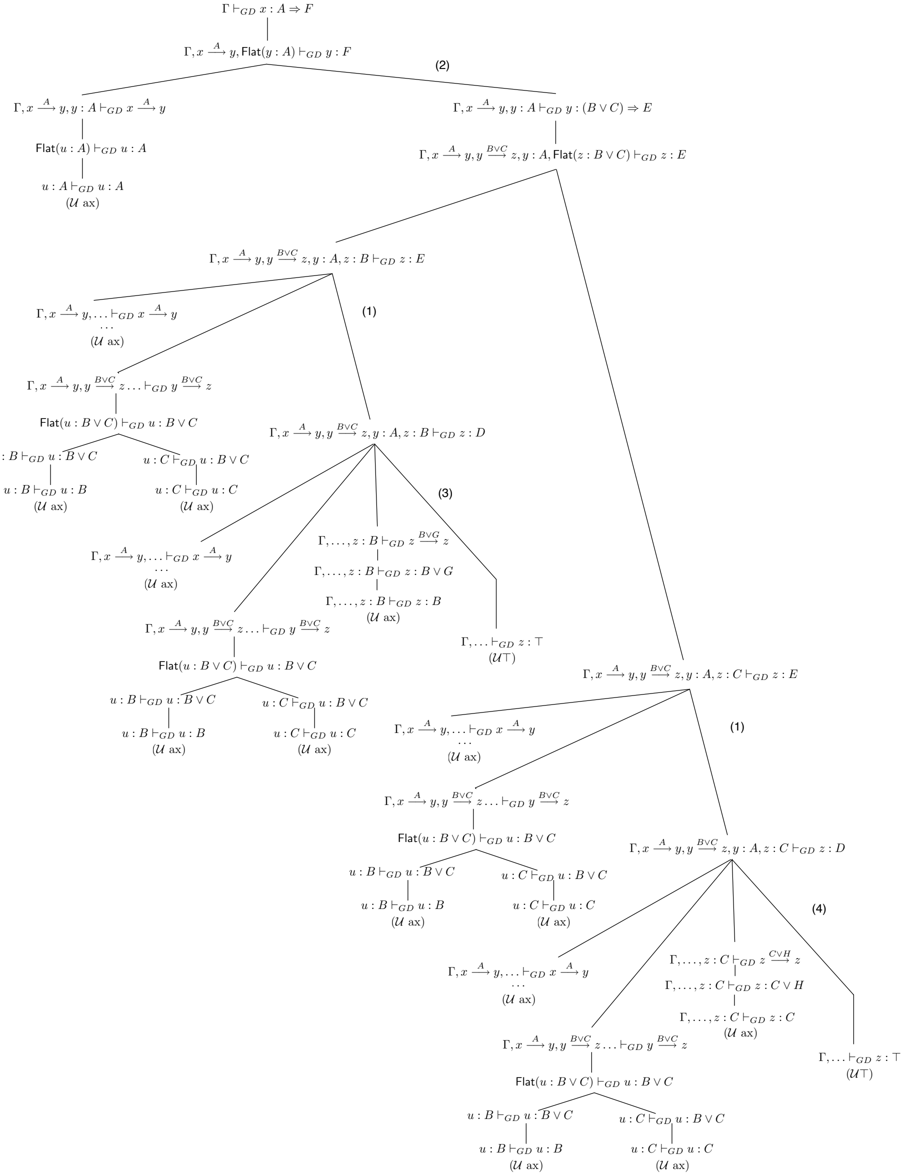

$\Gamma \vdash_{GD} x : A \Rightarrow F$
$\Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash_{GD} y : F$
(2)
$\Gamma, x \xrightarrow{A} y, y : A \vdash_{GD} x \xrightarrow{A} y$
$\mathsf{Flat}(u : A) \vdash_{GD} u : A$
$u : A \vdash_{GD} u : A$
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y : A \vdash_{GD} y : (B \vee C) \Rightarrow E$
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z, y : A, \mathsf{Flat}(z : B \vee C) \vdash_{GD} z : E$
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z, y : A, z : B \vdash_{GD} z : E$
(1)
$\Gamma, x \xrightarrow{A} y, \dots \vdash_{GD} x \xrightarrow{A} y$
...
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z \dots \vdash_{GD} y \xrightarrow{B\vee C} z$
$\mathsf{Flat}(u : B \vee C) \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B$
($\mathcal{U}$ ax)
$u : C \vdash_{GD} u : B \vee C$
$u : C \vdash_{GD} u : C$
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z, y : A, z : B \vdash_{GD} z : D$
(3)
$\Gamma, x \xrightarrow{A} y, \dots \vdash_{GD} x \xrightarrow{A} y$
...
($\mathcal{U}$ ax)
$\Gamma, \dots, z : B \vdash_{GD} z \xrightarrow{B\vee G} z$
$\Gamma, \dots, z : B \vdash_{GD} z : B \vee G$
$\Gamma, \dots, z : B \vdash_{GD} z : B$
($\mathcal{U}$ ax)
$\Gamma, \dots \vdash_{GD} z : \top$
($\mathcal{U}\top$)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z \dots \vdash_{GD} y \xrightarrow{B\vee C} z$
$\mathsf{Flat}(u : B \vee C) \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B$
($\mathcal{U}$ ax)
$u : C \vdash_{GD} u : B \vee C$
$u : C \vdash_{GD} u : C$
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z, y : A, z : C \vdash_{GD} z : E$
(1)
$\Gamma, x \xrightarrow{A} y, \dots \vdash_{GD} x \xrightarrow{A} y$
...
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z \dots \vdash_{GD} y \xrightarrow{B\vee C} z$
$\mathsf{Flat}(u : B \vee C) \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B$
($\mathcal{U}$ ax)
$u : C \vdash_{GD} u : B \vee C$
$u : C \vdash_{GD} u : C$
($\mathcal{U}$ ax)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z, y : A, z : C \vdash_{GD} z : D$
(4)
$\Gamma, x \xrightarrow{A} y, \dots \vdash_{GD} x \xrightarrow{A} y$
...
($\mathcal{U}$ ax)
$\Gamma, \dots, z : C \vdash_{GD} z \xrightarrow{C\vee H} z$
$\Gamma, \dots, z : C \vdash_{GD} z : C \vee H$
$\Gamma, \dots, z : C \vdash_{GD} z : C$
($\mathcal{U}$ ax)
$\Gamma, \dots \vdash_{GD} z : \top$
($\mathcal{U}\top$)
$\Gamma, x \xrightarrow{A} y, y \xrightarrow{B\vee C} z \dots \vdash_{GD} y \xrightarrow{B\vee C} z$
$\mathsf{Flat}(u : B \vee C) \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B \vee C$
$u : B \vdash_{GD} u : B$
($\mathcal{U}$ ax)
$u : C \vdash_{GD} u : B \vee C$
$u : C \vdash_{GD} u : C$
($\mathcal{U}$ ax)

Figure 4.10: A derivation in $\mathcal{U}$ID+MP of the goal $x : A \Rightarrow F$. When **($\mathcal{U}$ prop)** is applied, the corresponding edge is labeled with the number of the formula of the initial database used.

4.7 Conclusions

In this chapter we have provided a labelled calculus for minimal conditional logic CK, and its standard extensions with conditions ID, MP, CS and CEM. We have found cut-free and analytic systems for almost all studied systems, except for those presenting both MP and CEM. By refining these calculi, we have obtained a decision procedure for the respective logics. Moreover, we have been able to show that these logics are PSPACE. To the best of our knowledge, sequent calculi for these logics have not been previously studied and the complexity bound for them is new. As mentioned in the Introduction, most of the other works in the literature have concentrated exclusively on extensions of CK. For a broader discussion on related works, see Section 1.4.1.

Furthermore, we have presented a tighter space complexity bound for CK{+ID} which is based on the disjunction property of conditional formulas. We have also begun the investigation of a goal directed proof procedure for these conditional logics in the style of Miller's uniform proofs.

Chapter 5

Analytic Tableau Calculi for KLM Logics

In this chapter we introduce tableau procedures for all KLM logics; these calculi are called $\mathcal{T}K$, where K stands for **R**, **P**, **CL**, and **C**. For the purpose of exposition, we present first the calculus for **P**, which is the simplest one. Then we present the calculi for **CL**, **C**, and **R**. We can use the tableau calculi to define a decision procedure and obtain also a complexity bound for these logics, namely that they are all **coNP**-complete with the exception of **C**. In case of **CL** this bound is new, to the best of our knowledge.

Preliminary results of this chapter have already been presented in [GGOP05b, GGOP05a, GGOP06b, GGOP06a, GGOP09b].

5.1 Introduction

In Chapter 3 we have introduced KLM logics. As we have mentioned, even if KLM was born as an inferential approach to nonmonotonic reasoning, curiously enough, there has not been much investigation on deductive mechanisms for these logics. The state of the art has been discussed in Section 1.4.2.

In this work we introduce tableau procedures for all KLM logics, starting with the preferential logic **P**. Our approach is based on a novel interpretation of **P** into modal logics. As a difference with previous approaches (e.g. Crocco and Lamarre [CL92] and Boutillier [Bou94]), that take S4 as the modal counterpart of **P**, we consider here Gödel-Löb modal logic of provability G (see for instance [HC84]).

Our tableau method provides a sort of run-time translation of **P** into modal logic G. The idea is simply to interpret the preference relation as an accessibility relation: a conditional $A \mid\sim B$ holds in a model if B is true in all minimal A-worlds w (i.e. worlds in which A holds and that are minimal). An A-world w is a minimal A-world if all smaller worlds are not A-worlds. The relation with modal logic G is motivated by the fact that we assume, following KLM, the so-called *smoothness condition*, which is related to the well-known *limit assumption*. This condition ensures that minimal A-worlds exist whenever there are A-worlds, by preventing infinitely descending chains of worlds. This condition therefore corresponds to the finite-chain condition on the accessibility relation (as in modal logic G). Therefore, our interpretation of conditionals is different from the one proposed by Boutilier, who rejects the smoothness condition and then gives a less natural (and more complicated) interpretation of **P** into modal logic S4.

As a further difference with previous approaches, we do not give a formal translation of **P** into G. Rather, we directly provide a tableau calculus for **P**. One can notice some similarities between some of the rules for **P** and some of the rules for G. This is due to the correspondence between the semantics of the two logics. For deductive purposes, we believe that our approach is more direct, intuitive, and efficient than translating **P** into G and then using a calculus for G.

We are able to extend our approach to the cases of **CL** and **C** by using a second modality which takes care of states. Regarding **CL**, we show that we can map **CL**-models into **P**-models with an additional modality. The very fact that one can interpret **CL** into **P** by means of an additional modality does not seem to be previously known and might be of independent interest. In both cases, **P** and **CL**, we can define a decision procedure and obtain also a complexity bound for these logics, namely that they are both **coNP**-complete. In case of **CL** this bound is new, to the best of our knowledge.

We treat **C** in a similar way: we can establish a mapping between Cumulative models and a kind of bi-modal models. However, because of the lack of transitivity, the target modal logic is no longer G. The reason is that the *smoothness condition* (for any formula A, if a state satisfies A, then either it is minimal or it admits a smaller minimal state satisfying A) can no longer be identified with the finite-chain condition of G. As a matter of fact, the smoothness condition for **C** cannot be identified with any property of the accessibility relation, as it involves unavoidably the evaluation of formulas in worlds. We can still derive a tableau calculus based on our semantic mapping. But we pay a price: as a difference with **P** and **CL** the calculus for **C** requires a sort of (analytic) cut rule to account for the smoothness condition. This calculus gives nonetheless a decision procedure for **C**.

Finally, we consider the case of the strongest logic **R**; as for the other weaker systems, our approach is based on an interpretation of **R** into an extension of modal logic G, including modularity of the preference relation (previous approaches [CL92, Bou94] take S4.3 as the modal counterpart of **R**). As a difference with the tableau calculi introduced for **P**, **CL**, and **C**, here we develop a *labelled* tableau system, which seems to be the most natural approach in order to capture the modularity of the preference relation. The calculus defines a systematic procedure which allows the satisfiability problem for **R** to be decided in nondeterministic polynomial time, in accordance with the known complexity results for this logic.

From the completeness of our calculi we get for free the finite model property for all the logics considered. With the exception of the calculus for **C**, in order to ensure termination, our tableau procedures for KLM logics do not need any loop-checking, nor blocking, nor caching machinery. Termination is guaranteed only by adopting a restriction on the order of application of the rules.

In the following sections we present the tableaux calculi for the logics introduced. As mentioned, we start by presenting the calculus for **P**, which is the simpler and more general one. The calculi for **CL**, **C**, and **R** will become more understandable once the calculus for **P** is known.

$$(\mathbf{AX})\ \Gamma, P, \neg P \quad \text{with } P \in ATM \qquad (\neg)\ \frac{\Gamma, \neg\neg F}{\Gamma, F} \qquad (\wedge^{+})\ \frac{\Gamma, F \wedge G}{\Gamma, F, G}$$

$$(\wedge^{-})\ \frac{\Gamma, \neg(F \wedge G)}{\Gamma, \neg F \qquad \Gamma, \neg G} \qquad (\vee^{+})\ \frac{\Gamma, F \vee G}{\Gamma, F \qquad \Gamma, G} \qquad (\vee^{-})\ \frac{\Gamma, \neg(F \vee G)}{\Gamma, \neg F, \neg G}$$

$$(\rightarrow^{+})\ \frac{\Gamma, F \rightarrow G}{\Gamma, \neg F \qquad \Gamma, G} \qquad (\rightarrow^{-})\ \frac{\Gamma, \neg(F \rightarrow G)}{\Gamma, F, \neg G}$$

$$(\mid\!\sim^{+})\ \frac{\Gamma, A \mid\!\sim B}{\Gamma, A \mid\!\sim B, \neg A \qquad \Gamma, A \mid\!\sim B, \neg\Box\neg A \qquad \Gamma, A \mid\!\sim B, B}$$

$$(\mid\!\sim^{-})\ \frac{\Gamma, \neg(A \mid\!\sim B)}{\Gamma^{\mid\sim^{\pm}}, A, \Box\neg A, \neg B} \qquad (\Box^{-})\ \frac{\Gamma, \neg\Box\neg A}{\Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\sim^{\pm}}, A, \Box\neg A}$$

Figure 5.1: Tableau system $\mathcal{TP}$.

5.2 A Tableau Calculus for Preferential Logic $\mathbf{P}$

In this section we present a tableau calculus for **P** called $\mathcal{TP}$. We then analyze it in order to give an optimal decision procedure for **P**, matching the known complexity results for this logic.

As already mentioned in Section 3.2.2, we consider the language $\mathcal{L}_P$, which extends $\mathcal{L}$ by boxed formulas of the form $\Box\neg A$.

Definition 5.1 (The calculus $\mathcal{TP}$). *The rules of the calculus manipulate sets of formulas Γ. We write Γ, F as a shorthand for $\Gamma \cup \{F\}$. Moreover, given Γ we define the following sets:*

- $\Gamma^{\Box} = \{\Box\neg A \mid \Box\neg A \in \Gamma\}$
- $\Gamma^{\Box^{\downarrow}} = \{\neg A \mid \Box\neg A \in \Gamma\}$
- $\Gamma^{\mid\sim^{+}} = \{A \mid\!\sim B \mid A \mid\!\sim B \in \Gamma\}$
- $\Gamma^{\mid\sim^{-}} = \{\neg(A \mid\!\sim B) \mid \neg(A \mid\!\sim B) \in \Gamma\}$
- $\Gamma^{\mid\sim^{\pm}} = \Gamma^{\mid\sim^{+}} \cup \Gamma^{\mid\sim^{-}}$

The tableau rules are given in Figure 5.1. A tableau is a tree whose nodes are sets of formulas Γ. A branch is a sequence of nodes $\Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$ Each node Γ_i is obtained from its immediate predecessor Γ_{i-1} by applying a rule of $\mathcal{TP}$, having Γ_{i-1} as the premise and Γ_i as one of its conclusions. A branch is closed if one of its nodes is an instance of (**AX**)*, otherwise it is open. We say that a tableau is closed if all its branches are closed. A node Γ is consistent if no tableau for Γ is closed. We call* tableau for Γ *a tableau having Γ as a root.*

In order to check whether a set of formulas Γ is unsatisfiable, we verify if there is a closed tableau for Γ. First of all, we distinguish between *static* and *dynamic* rules.

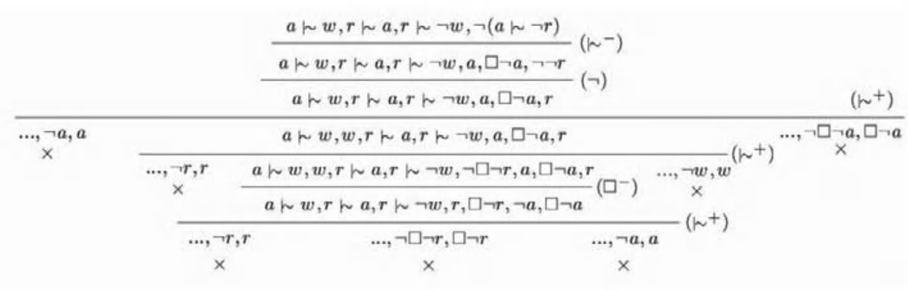

a |~ w, r |~ a, r |~ ¬w, ¬(a |~ ¬r)
(|~⁻)
a |~ w, r |~ a, r |~ ¬w, a, □¬a, ¬¬r
(¬)
a |~ w, r |~ a, r |~ ¬w, a, □¬a, r
(|~⁺)
..., ¬a, a
×
a |~ w, w, r |~ a, r |~ ¬w, a, □¬a, r
..., ¬□¬a, □¬a
×
(|~⁺)
..., ¬r, r
×
a |~ w, w, r |~ a, r |~ ¬w, ¬□¬r, a, □¬a, r
..., ¬w, w
×
(□⁻)
a |~ w, r |~ a, r |~ ¬w, r, □¬r, ¬a, □¬a
(|~⁺)
..., ¬r, r
×
..., ¬□¬r, □¬r
×
..., ¬a, a
×

Figure 5.2: A derivation of *adult* $\mid\!\sim$ *worker*, *retired* $\mid\!\sim$ *adult*, *retired* $\mid\!\sim$ $\neg$*worker*, $\neg$(*adult* $\mid\!\sim$ $\neg$*retired*). For readability, we use a to denote *adult*, r for *retired*, and w for *worker*.

The rules ($\mid\!\sim^-$) and ($\Box^-$) are called *dynamic*, since their conclusions represent another world with respect to the one represented by the premise; the other rules are called *static*, since the world represented by premise and conclusion(s) is the same. The rules for the boolean propositions are the usual ones. According to the rule ($\mid\!\sim^-$), if a negated conditional $\neg(A \mid\!\sim B)$ holds in a world, then there is a minimal A-world (i.e. in which A and $\Box\neg A$ hold) which falsifies B. According to the rule ($\mid\!\sim^+$), if a positive conditional $A \mid\!\sim B$ holds in a world, then either the world falsifies A or it is not minimal for A (i.e. $\neg\Box\neg A$ holds) or it is a B-world. According to the rule ($\Box^-$), if a world satisfies $\neg\Box\neg A$, by the strong smoothness condition there must be a preferred minimal A-world, i.e. a world in which A and $\Box\neg A$ hold.

In our calculus $\mathcal{T}\mathbf{P}$, axioms are restricted to atomic formulas only. It is easy to extend axioms to a generic formula F, as stated by the following proposition:

Proposition 5.2. *Given a formula F and a set of formulas Γ, then $\Gamma, F, \neg F$ has a closed tableau.*

Proof. By an easy inductive argument on the structure of the formula F. The proof is easy and left to the reader.

As an example, we show that *adult* $\mid\!\sim$ $\neg$*retired* can be inferred from a knowledge base containing the following assertions: *adult* $\mid\!\sim$ *worker*, *retired* $\mid\!\sim$ *adult*, *retired* $\mid\!\sim$ $\neg$*worker*. Figure 5.2 shows a derivation for the initial node *adult* $\mid\!\sim$ *worker*, *retired* $\mid\!\sim$ *adult*, *retired* $\mid\!\sim$ $\neg$*worker*, $\neg$(*adult* $\mid\!\sim$ $\neg$*retired*).
Our tableau calculus $\mathcal{T}\mathbf{P}$ is based on a runtime translation of conditional assertions into modal logic G. As we have seen in Section 3.2.2, this allows a characterization of the minimal worlds satisfying a formula A (i.e., the worlds in $Min_<(A)$) as the worlds w satisfying the formula $A \land \Box\neg A$. It is tempting to provide a full translation of the conditionals in the logic G, and then to use the standard tableau calculus for G. To this purpose, we can exploit the transitivity properties of G frames in order to capture the fact that conditionals are global to all worlds by the formula $\Box(A \land \Box\neg A \rightarrow B)$. Hence, the overall translation of a conditional formula $A \mid\!\sim B$ could be the following one: $(A \land \Box\neg A \rightarrow B) \land \Box(A \land \Box\neg A \rightarrow B)$. However, there would be significant differences between the calculus resulting from the translation and our calculus.

Using the standard tableau rules for G on the translation, we would get the rule ($\mid\!\sim^+$) as a derived rule. Instead, the rule for dealing with negated conditionals (which would be translated in G as a disjunction of two formulas, namely $(A \wedge \Box\neg A \wedge \neg B) \vee \Diamond(A \wedge \Box\neg A \wedge \neg B))$, would be rather different.

Let us first observe that the rule $(\mid\!\sim^-)$ we have introduced precisely captures the intuition that conditionals are global, hence (1) all conditionals are kept in the conclusion of the rule and (2) when moving to a new minimal world, all the boxed formulas (positive and negated) are removed. Conversely, when the tableau rules for G are applied to the translation of the negated conditionals, we get two branches (due to the disjunction). None of the branches can be eliminated. In both branches all the boxed formulas are kept, while negated conditionals are erased. This is quite different from our rule $(\mid\!\sim^-)$, and it is not that obvious that the calculus obtained by the translation of **P** conditionals in G is equivalent to $\mathcal{T}$**P**. Roughly speaking, point (2) can be explained as follows: when a negated conditional $\neg(A \mid\!\sim B)$ is evaluated in a world w, this corresponds to finding a minimal A-world w' satisfying $\neg B$ (a world satisfying $A, \Box\neg A, \neg B$). w' does not depend from w (since conditionals are global), hence boxed formulas, keeping information about w, can be removed.

Also observe that, from the semantic point of view, the model extracted from an open tableau has the structure of a forest, while the model constructed by applying the tableau for G to the translation of conditionals has the structure of a tree. This difference is due to the fact that the above translation of **P** in G uses the same modality $\Box$ both for capturing the minimality condition and for modelling the fact that conditionals are global. For this reason, a translation to G as the one proposed above for **P**, would not be applicable to the cumulative logic **C**, as the relation $<$ is not transitive in **C**. Moreover, the treatment of both the logics **C** and **CL** would anyhow require the addition to the language of a new modality to deal with states. The advantage of the runtime translation we have adopted is that of providing a uniform approach to deal with the different logics. Furthermore, in case of **P** the target logic G is well-established and clear, whereas for the other cases the target logic would be more complicated (as it may combine different modalities).

The system $\mathcal{T}$**P** is sound and complete with respect to the semantics. As usual, given a model $\mathcal{M}$, a world w, and a set of formulas Γ, we write $\mathcal{M}, w \models \Gamma$ as a shorthand for $\mathcal{M}, w \models F$ for all $F \in \Gamma$.

Theorem 5.3 (Soundness of $\mathcal{T}$**P**)**.** *The system $\mathcal{T}$**P** is sound with respect to preferential models, i.e. if there is a closed tableau for a set Γ, then Γ is unsatisfiable.*

Proof. As usual, we proceed by induction on the structure of the closed tableau having Γ as a root. The base case is when the tableau consists of a single node; in this case, both P and $\neg P$ occur in Γ, therefore Γ is obviously unsatisfiable. For the inductive step, we have to show that, for each rule r, if all the conclusions of r are unsatisfiable, then the premise is unsatisfiable too. We show the contrapositive, i.e. we prove that if the premise of r is satisfiable, then at least one of the conclusions is satisfiable. As usual, a node Γ is satisfiable if there is a model $\mathcal{M}$ and a world w such that $\mathcal{M}, w \models \Gamma$. Boolean cases are easy and left to the reader. We present the cases for conditional and box rules:

- $(\mid\!\sim^+)$: if $\Gamma, A \mid\!\sim B$ is satisfiable, then there exists a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ with some world $w \in \mathcal{W}$ such that $\mathcal{M}, w \models \Gamma, A \mid\!\sim B$. We distinguish the two following cases:
  - $\mathcal{M}, w \not\models A$, thus $\mathcal{M}, w \models \neg A$: in this case, the left conclusion of the $(\mid\!\sim^+)$ rule is satisfied $(\mathcal{M}, w \models \Gamma, A \mid\!\sim B, \neg A)$;

  - $\mathcal{M}, w \models A$: we consider two subcases:
    - $w \in Min_<(A)$: by the definition of $\mathcal{M}, w \models A \mid\sim B$, we have that for all $w' \in Min_<(A)$, $\mathcal{M}, w' \models B$. Therefore, we have that $\mathcal{M}, w \models B$ and the right conclusion of $(\mid\sim^+)$ is satisfiable;
    - $w \notin Min_<(A)$: by the smoothness condition, there exists a world $w' < w$ such that $w' \in Min_<(A)$; therefore, $\mathcal{M}, w \models \neg\Box\neg A$ by the definition of $\Box$. The conclusion in the middle of the $(\mid\sim^+)$ rule is then satisfiable.

- $(\mid\sim^-)$: if $\Gamma, \neg(A \mid\sim B)$ is satisfiable, then $\mathcal{M}, w \models \Gamma$ and $(*)\mathcal{M}, w \not\models A \mid\sim B$ for some world w. By $(*)$, there is a world w' in the model $\mathcal{M}$ such that $w' \in Min_<(A)$ (i.e. $(1)\mathcal{M}, w' \models A$ and $(2)\mathcal{M}, w' \models \Box\neg A$) and $(3)\mathcal{M}, w' \not\models B$. By $(1), (2)$ and (3), we have that $\mathcal{M}, w' \models A, \Box\neg A, \neg B$. We conclude $\mathcal{M}, w' \models A, \Box\neg A, \neg B, \Gamma^{\mid\sim^\pm}$, since conditionals are "global" in a model.
- $(\Box^-)$: if $\Gamma, \neg\Box\neg A$ is satisfiable, then there is a model $\mathcal{M}$ and some world w such that $\mathcal{M}, w \models \Gamma, \neg\Box\neg A$, then $\mathcal{M}, w \not\models \Box\neg A$. By the truth definition of $\Box$, there exists a world w' such that $w' < w$ and $\mathcal{M}, w' \models A$. By the Strong Smoothness Condition, we can assume that w' is a *minimal* A-world. Therefore, $\mathcal{M}, w' \models \Box\neg A$ by the truth definition of $\Box$. It is easy to conclude that $\mathcal{M}, w' \models A, \Box\neg A, \Gamma^{\mid\sim^\pm}, \Gamma^{\Box^\downarrow}, \Gamma^{\Box}$, as follows: 1. conditionals are global in a model (hence $\mathcal{M}, w' \models \Gamma^{\mid\sim^\pm}$); 2. formulas in $\Gamma^{\Box^\downarrow}$ are true in w' since $w' < w$ and 3. the $<$ relation is transitive, thus boxed formulas holding in w (i.e. $\Gamma^{\Box}$) also hold in w'.

■

To prove the completeness of $\mathcal{TP}$ we have to show that if Γ is unsatisfiable, then there is a closed tableau starting with Γ. The proof is inspired by [Gor99]. We prove the contrapositive, that is: if there is no closed tableau for Γ, then there is a model satisfying Γ. We first introduce the *saturation* of a tableau node Γ. Intuitively, given a node Γ, we say that it is saturated if the following condition holds: if the node contains the premise of a static rule, then it also contains one of its conclusions.

Definition 5.4 (Saturated node)**.** *A tableau node Γ is saturated with respect to the static rules if the following conditions hold:*

- *if $F \wedge G \in \Gamma$ then $F \in \Gamma$ and $G \in \Gamma$;*
- *if $\neg(F \wedge G) \in \Gamma$ then $\neg F \in \Gamma$ or $\neg G \in \Gamma$;*
- *if $F \vee G \in \Gamma$ then $F \in \Gamma$ or $G \in \Gamma$;*
- *if $\neg(F \vee G) \in \Gamma$ then $\neg F \in \Gamma$ and $\neg G \in \Gamma$;*
- *if $F \rightarrow G \in \Gamma$ then $\neg F \in \Gamma$ or $G \in \Gamma$;*
- *if $\neg(F \rightarrow G) \in \Gamma$ then $F \in \Gamma$ and $\neg G \in \Gamma$;*
- *if $\neg\neg F \in \Gamma$ then $F \in \Gamma$;*
- *if $A \mid\sim B \in \Gamma$ then $\neg A \in \Gamma$ or $\neg\Box\neg A \in \Gamma$ or $B \in \Gamma$.*

It is easy to observe that the following Lemma holds:

Lemma 5.5. *Given a consistent finite set of formulas Γ, there is a consistent, finite, effectively computable, and saturated node Γ' such that $\Gamma' \supseteq \Gamma$.*

Proof. Consider the set $\Gamma^\bigstar$ of complex formulas in Γ such that there is a static rule that has not yet been applied to that formula in Γ. For instance, if $\Gamma = \{P \mid\!\sim Q, \neg(R \rightarrow S), R, \neg S, R \vee T, \neg\Box\neg Q\}$, then $\Gamma^\bigstar = \{P \mid\!\sim Q\}$, since neither $\neg P$ nor $\neg\Box\neg P$ nor Q, resulting from an application of the static rule $(\mid\!\sim^+)$ to $P \mid\!\sim Q$, belong to Γ.

If $\Gamma^\bigstar$ is empty, we are done. Otherwise, we construct the saturated set Γ' as follows: 1. initialize Γ' with Γ; 2. choose a complex formula F in $\Gamma^\bigstar$ and apply the *static* rule corresponding to its principal operator; 3. add to Γ' the formula(s) of (one of) the *consistent* conclusions obtained by applying the static rule; 4. update $\Gamma^\bigstar$[1] and repeat from 2. until $\Gamma^\bigstar$ is empty. This procedure terminates, since in all static rules the conclusions have a lower complexity than the premise; a brief discussion on the $(\mid\!\sim^+)$ rule: from the premise $A \mid\!\sim B$ we have the following possible conclusions: $\neg A$, $\neg\Box\neg A$ and B. If $\neg\Box\neg A$ is introduced, then no other static rule will be applied to it. Since A and B are boolean combinations of formulas, then the other applications of static rules to them will decrease the complexity.

Furthermore, each step of the procedure preserves the consistency of Γ. Indeed, each conclusion of the rule applied to Γ corresponds to a branch of a tableau for Γ. If all the branches were inconsistent, Γ would have a closed tableau, hence would be inconsistent, against the hypothesis.

■

By Lemma 5.5, we can think of having a function which, given a consistent set of formulas Γ, returns one fixed consistent saturated node, denoted by $\texttt{SAT}(\Gamma)$. Moreover, we denote by $\texttt{APPLY}(\Gamma, R, F)$ the result of applying the tableau rule (R) to Γ, with principal formula F. In case (R) has several conclusions (the case of a branching), we suppose that the function APPLY chooses one consistent conclusion in an arbitrary but fixed manner.

Theorem 5.6 (Completeness of $\mathcal{TP}$). *$\mathcal{TP}$ is complete with respect to preferential models, i.e. if a set of formulas Γ is unsatisfiable, then Γ has a closed tableau in $\mathcal{TP}$.*

Proof. We assume that no tableau for Γ is closed, then we construct a model for Γ. We build X, the set of worlds of the model, as follows:

```
1. initialize $X = \{\texttt{SAT}(\Gamma)\}$; mark $\texttt{SAT}(\Gamma)$ as unresolved;
while $X$ contains unresolved nodes do
    2. choose an unresolved $\Gamma_i$ from $X$;
    3. for each formula $\neg(A \mid\!\sim B) \in \Gamma_i$
        3a. let $\Gamma_{\neg(A \mid\sim B)} = \texttt{SAT}(\texttt{APPLY}(\Gamma_i, \mid\!\sim^-, \neg(A \mid\!\sim B)))$;
        3b. if $\Gamma_{\neg(A \mid\sim B)} \not\in X$ then $X = X \cup \{\Gamma_{\neg(A \mid\sim B)}\}$ and mark $\Gamma_{\neg(A \mid\sim B)}$ as unresolved;
    4. for each formula $\neg\Box\neg A \in \Gamma_i$, let $\Gamma_{\neg\Box\neg A} = \texttt{SAT}(\texttt{APPLY}(\Gamma_i, \Box^-, \neg\Box\neg A))$;
        4a. add the relation $\Gamma_{\neg\Box\neg A} < \Gamma_i$;
        4b. if $\Gamma_{\neg\Box\neg A} \not\in X$ then $X = X \cup \{\Gamma_{\neg\Box\neg A}\}$ and mark $\Gamma_{\neg\Box\neg A}$ as unresolved;
    5. mark $\Gamma_i$ as resolved;
endWhile;
```

This procedure terminates, since the number of possible nodes that can be obtained by applying $\mathcal{TP}$'s rules to an initial finite set Γ is finite. We construct the model $\mathcal{M} = \langle X, <_X, V\rangle$ for Γ as follows:

[1] The complex formula analyzed at the current step must be removed from $\Gamma^\bigstar$ and formulas obtained by the application of the static rules that fulfill the definition of $\Gamma^\bigstar$ must be added.

- $<_X$ is the transitive closure of the relation $<$;
- for each $\Gamma_i \in X$, we have $V(\Gamma_i) = \{P \mid P \in \Gamma_i \cap ATM\}$

In order to show that $\mathcal{M}$ is a preferential model for Γ, we prove the following facts:

Fact 5.7. *The relation $<_X$ is acyclic.*

Proof of Fact 5.7. Suppose on the contrary there is a loop $\Gamma_i < \cdots < \Gamma_j < \Gamma_i$. Each $<$ has been introduced by step 4 in the procedure above. Γ_j has been obtained by an application of $(\Box^-)$ to a given $\neg\Box\neg A \in \Gamma_i$. This entails that, for all $\Gamma_k <_X \Gamma_i$, we have that $\Box\neg A \in \Gamma_k$, hence $\Box\neg A \in \Gamma_i$, which contradicts the fact that Γ_i is consistent.

□ *Fact 5.7*

Fact 5.8. *The relation $<_X$ is irreflexive, transitive, and satisfies the smoothness condition.*

Proof of Fact 5.8. Transitivity follows by construction. Irreflexivity follows the acyclicity, since $\Gamma_i <_X \Gamma_i$ is never added. As there are finitely many worlds, and the relation $<_X$ is acyclic, it follows that there cannot be infinite descending chains. This fact, together with the transitivity of $<_X$, entails that $<_X$ satisfies the smoothness condition.

□ *Fact 5.8*

The only rules introducing a new world in X in the procedure above are $(\mid\!\sim^-)$ and $(\Box^-)$. Since these two rules keep positive conditionals in their conclusions, it follows that any positive conditional $A \mid\!\sim B$ belonging to SAT(Γ), where Γ is the initial set of formulas, also belongs to each world introduced in X. This is stated in a rigorous way as follows:

Fact 5.9. *Given a world $\Gamma_i \in X$ and any positive conditional $A \mid\!\sim B$, we have that $A \mid\!\sim B \in \Gamma_i$ iff $A \mid\!\sim B \in$ SAT(Γ).*

Proof of Fact 5.9. It can be shown that *only* the conditionals in SAT(Γ) belong to possible worlds in X. Indeed, all worlds in X are generated by the application of a dynamic rule, followed by the application of static rules for saturation. It can be shown that this combination of rules does never introduce a new conditional.

□ *Fact 5.9*

We conclude by proving the following fact:

Fact 5.10. *For all formulas F and for all worlds $\Gamma_i \in X$ we have that:*
(i) if $F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \models F$; (ii) if $\neg F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \not\models F$.

Proof of Fact 5.10. By induction on the structure of F. If F is an atom P, then $P \in \Gamma_i$ implies $\mathcal{M}, \Gamma_i \models P$ by definition of V. Moreover, $\neg P \in \Gamma_i$ implies that $P \not\in \Gamma_i$ as Γ_i is consistent; thus, $\mathcal{M}, \Gamma_i \not\models P$ (by definition of V). For the inductive step we only consider the cases of positive and negative box formulas and of positive and negative conditional formulas:

- $\Box\neg A \in \Gamma_i$. Then, for all $\Gamma_j <_X \Gamma_i$ we have $\neg A \in \Gamma_j$ by definition of $(\Box^-)$, since Γ_j has been generated by a sequence of applications of $(\Box^-)$. By inductive hypothesis $\mathcal{M}, \Gamma_j \not\models A$ for all $\Gamma_j <_X \Gamma_i$, whence $\mathcal{M}, \Gamma_i \models \Box\neg A$.
- $\neg\Box\neg A \in \Gamma_i$. By construction there is a $\Gamma_j \in X$ s.t. $\Gamma_j <_X \Gamma_i$ and $A \in \Gamma_j$. By inductive hypothesis $\mathcal{M}, \Gamma_j \models A$. Thus, $\mathcal{M}, \Gamma_i \not\models \Box\neg A$.
- $A \mid\!\sim B \in \Gamma_i$. Let $\Gamma_j \in Min_{<_X}(A)$; one can observe that (1)$\neg A \in \Gamma_j$ or (2)$\neg\Box\neg A \in \Gamma_j$ or (3)$B \in \Gamma_j$, since $A \mid\!\sim B \in \Gamma_j$ by Fact 5.9, and since Γ_j is saturated. (1) cannot be the case, since otherwise by inductive hypothesis $\mathcal{M}, \Gamma_j \not\models A$, which contradicts the definition of $Min_{<_X}(A)$. If (2), by construction of $\mathcal{M}$ there exists a node $\Gamma_k <_X \Gamma_j$ such that $A \in \Gamma_k$: by inductive hypothesis $\mathcal{M}, \Gamma_k \models A$, which contradicts $\Gamma_j \in Min_{<_X}(A)$. Thus it must be that (3)$B \in \Gamma_j$, and by inductive hypothesis $\mathcal{M}, \Gamma_j \models B$. Hence, we can conclude $\mathcal{M}, \Gamma_i \models A \mid\!\sim B$.
- $\neg(A \mid\!\sim B) \in \Gamma_i$. By construction of X, there exists $\Gamma_j \in X$ such that $A, \Box\neg A, \neg B \in \Gamma_j$. By inductive hypothesis we have that $\mathcal{M}, \Gamma_j \models A$ and $\mathcal{M}, \Gamma_j \models \Box\neg A$. It follows that $\Gamma_j \in Min_{<_X}(A)$. Furthermore, always by induction, $\mathcal{M}, \Gamma_j \not\models B$. Hence, $\mathcal{M}, \Gamma_i \not\models A \mid\!\sim B$.

□ *Fact 5.10*

By the above Facts the proof of the completeness of $\mathcal{T}$**P** is over, since $\mathcal{M}$ is a model for the initial set Γ.

■

The construction of the model above provides a constructive proof of the finite model property of **P**:

Corollary 5.11 (Finite model property)**. P** *has the finite model property.*

Proof. By Theorem 5.3, if Γ is satisfiable, then there is no closed tableau for Γ. By the construction in the proof of Theorem 5.6, if there is no closed tableau for Γ, then Γ is satisfiable in a finite model.

■

A relevant property of the calculus that will be useful to estimate the complexity of logic **P** is the so-called *disjunction property* of conditional formulas. The reason why this property holds is that the $(\mid\!\sim^-)$ rule discards all the other formulas that could have been introduced by its previous application.

Proposition 5.12 (Disjunction property)**.** *If there is a closed tableau for* $\Gamma, \neg(A \mid\!\sim B), \neg(C \mid\!\sim D)$*, then there is a closed tableau either for* $\Gamma, \neg(A \mid\!\sim B)$ *or for* $\Gamma, \neg(C \mid\!\sim D)$*.*

Proof. Consider a closed tableau for $\Gamma, \neg(A \mid\!\sim B), \neg(C \mid\!\sim D)$. If the tableau does not contain any application of $(\mid\!\sim^-)$, then the property immediately follows. The same holds if either $\neg(A \mid\!\sim B)$ or $\neg(C \mid\!\sim D)$ are not used in the tableau. Consider the case in which there is an application of $(\mid\!\sim^-)$ first to $\neg(C \mid\!\sim D)$, and then to $\neg(A \mid\!\sim B)$. We show that in this case there is also a closed tableau for $\Gamma, \neg(A \mid\!\sim B)$. We can build a tableau of the form:

$$
\begin{array}{cl}
\Gamma, \neg(A \mid\!\sim B), \neg(C \mid\!\sim D) & \\
\Pi_1 & \\
\Gamma', \neg(A \mid\!\sim B), \neg(C \mid\!\sim D) & \\
 & (\mid\!\sim^-) \\
\Gamma'^{\mid\sim^{\pm}}, \neg(A \mid\!\sim B), C, \Box\neg C, \neg D & \\
\Pi_2 & \\
\Gamma'', \neg(A \mid\!\sim B) & \\
 & (\mid\!\sim^-) \\
\Gamma''^{\mid\sim^{\pm}}, A, \Box\neg A, \neg B &
\end{array}
$$

Since C and D are propositional formulas, $C, \Box\neg C, \neg D$ and, eventually, their subformulas introduced by the application of some boolean rules in Π_2, will be removed by the application of $(\mid\!\sim^-)$ on $\neg(A \mid\!\sim B)$. Therefore, one can obtain a closed tableau of $\Gamma, \neg(A \mid\!\sim B)$ as follows:

$$
\begin{array}{cl}
\Gamma, \neg(A \mid\!\sim B) & \\
\Pi'_1 & \\
\Gamma', \neg(A \mid\!\sim B) & \\
\Pi'_2 & \\
\Gamma^*, \neg(A \mid\!\sim B) & \\
 & (\mid\!\sim^-) \\
\Gamma''^{\mid\sim^{\pm}}, A, \Box\neg A, \neg B &
\end{array}
$$

Π'_1 is obtained by removing $\neg(C \mid\!\sim D)$ from all the nodes of Π_1;Π'_2 is obtained by removing from Π_2 the application of rules on $C, \neg D$ and their subformulas. The symmetric case, corresponding to the case in which $(\mid\!\sim^-)$ is applied first on $\neg(A \mid\!\sim B)$ and then on $\neg(C \mid\!\sim D)$, can be proved in the same manner, thus we can conclude that $\Gamma, \neg(C \mid\!\sim D)$ has a closed tableau.

■

5.2.1 Decidability and Complexity of P

Terminating procedure for P

In general, non-termination in tableau calculi can be caused by two different reasons: 1. dynamic rules may generate infinitely-many worlds, creating infinite branches; 2. some rules copy their principal formula in the conclusion, and can thus be reapplied over the same formula without any control.

Concerning the first source of non-termination (point 1.), we can observe that infinitely-many worlds cannot be generated on a branch by $(\mid\!\sim^-)$ rule, since this rule can be applied only once to a given negated conditional on a branch. Nonetheless, the interplay between rules $(\mid\!\sim^+)$ and $(\Box^-)$ might lead to generate infinite branches. However, we show that this cannot occur, once we introduce the following standard restriction on the order of application of the rules.

Definition 5.13 (Restriction on the calculus)**.** *Building a tableau for a set of formulas* Γ*, the application of the* $(\Box^-)$ *rule must be postponed to the application of the propositional rules and to the verification that* Γ *is an instance of* (**AX**)*.*

It is easy to observe that, without the restriction above, point 1. could occur; for instance, consider the following trivial example, showing a branch of a tableau starting with $P \mid\!\sim Q$, with $P, Q \in ATM$:

$$\begin{array}{ccl}
P \mid\!\sim Q & & \\
 & & (\mid\!\sim^+) \\
\neg\Box\neg P, P \mid\!\sim Q & \dots & \\
 & & (\Box^-) \\
P, \Box\neg P, P \mid\!\sim Q & & \\
 & & (\mid\!\sim^+) \\
P, \Box\neg P, \neg\Box\neg P, P \mid\!\sim Q & \dots & \\
 & & (\Box^-) \\
(*)\neg P, P, \Box\neg P, P \mid\!\sim Q & & \\
 & & (\mid\!\sim^+) \\
\neg P, P, \Box\neg P, \neg\Box\neg P, P \mid\!\sim Q & \dots & \\
 & & (\Box^-) \\
\neg P, P, \Box\neg P, P \mid\!\sim Q & & \\
 & & (\mid\!\sim^+) \\
\neg P, P, \Box\neg P, \neg\Box\neg P, P \mid\!\sim Q & \dots & \\
\dots & &
\end{array}$$

In the above example, the $(\Box^-)$ rule is applied systematically before the other rules, thus generating an infinite branch. However, if the restriction in Definition 5.13 is adopted, as it is easy to observe, the procedure terminates at the step marked with $(*)$. Indeed, the test that $\neg P, P, \Box\neg P, P \mid\!\sim Q$ is an instance of the axiom (**AX**) succeeds before applying $(\Box^-)$ again, and the branch is considered to be closed.
As already mentioned, we show that, given the above restriction, the calculus $\mathcal{T}\mathbf{P}$ is not affected by the problems of point 1. Notice that this would not work in other systems (for instance, in K4 one needs a more sophisticated loop-checking as described in [HSZ96]). More in detail, we show (Lemma 5.19 and Theorem 5.20) that the interplay between $(\mid\!\sim^+)$ and $(\Box^-)$ does not generate branches containing infinitely-many worlds. Intuitively, the application of $(\Box^-)$ to a formula $\neg\Box\neg A$ (introduced by $(\mid\!\sim^+)$) adds the formula $\Box\neg A$ to the conclusion, so that $(\mid\!\sim^+)$ can no longer consistently introduce $\neg\Box\neg A$. This is due to the properties of $\Box$ in G, and would not hold if $\Box$ had weaker properties (e.g. K4 properties).

Concerning point 2. the above calculus $\mathcal{T}\mathbf{P}$ does not ensure a terminating proof search due to $(\mid\!\sim^+)$, which can be applied without any control. We ensure the termination by putting some constraints on $\mathcal{T}\mathbf{P}$. The intuition is as follows: one does not need to apply $(\mid\!\sim^+)$ on the same conditional formula $A \mid\!\sim B$ *more than once in the same world*, therefore we keep track of positive conditionals already used by moving them in an additional set Σ in the conclusions of $(\mid\!\sim^+)$, and restrict the application of this rule to unused conditionals only. The dynamic rules re-introduce formulas from Σ in order to allow further applications of $(\mid\!\sim^+)$ in the other worlds. This machinery is standard.

The terminating calculus $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ is presented in Figure 5.3. Observe that the tableau nodes are now pairs $\Gamma; \Sigma$. In order to prove that a set of formulas Γ is unsatisfiable, one has to check whether there is a closed tableau in $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ having $\Gamma; \emptyset$ as a root. As an example, here again we show a proof in $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ of the fact that *adult* $\mid\!\sim$ $\neg$*retired* can be inferred from the assertions *adult* $\mid\!\sim$ *worker*, *retired* $\mid\!\sim$ *adult*, *retired* $\mid\!\sim$ $\neg$*worker*, already presented in Figure 5.2 by means of the rules of $\mathcal{T}\mathbf{P}$.

The calculus $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ is sound and complete with respect to the semantics:

Theorem 5.14 (Soundness and completeness of $\mathcal{T}\mathbf{P}^{\mathbf{T}}$). *Given a set of formulas Γ, it is unsatisfiable iff $\Gamma; \emptyset$ has a closed tableau in $\mathcal{T}\mathbf{P}^{\mathbf{T}}$.*

Proof. The soundness is immediate and left to the reader. The completeness easily follows from the fact that two applications of $(\mid\!\sim^+)$ to the same conditional in the same world are useless. Indeed, given a proof in $\mathcal{T}\mathbf{P}$, if $(\mid\!\sim^+)$ is applied *twice* to $\Gamma, A \mid\!\sim B$ in the same world, then we can assume, without loss of generality, that the

$$(\mathbf{AX})\ \Gamma, P, \neg P; \Sigma \quad \text{with } P \in ATM \qquad (\neg)\frac{\Gamma, \neg\neg F; \Sigma}{\Gamma, F; \Sigma} \qquad (\wedge^+)\frac{\Gamma, F \wedge G; \Sigma}{\Gamma, F, G; \Sigma}$$

$$(\wedge^-)\frac{\Gamma, \neg(F \wedge G); \Sigma}{\Gamma, \neg F; \Sigma \qquad \Gamma, \neg G; \Sigma} \qquad (\vee^+)\frac{\Gamma, F \vee G; \Sigma}{\Gamma, F; \Sigma \qquad \Gamma, G; \Sigma} \qquad (\vee^-)\frac{\Gamma, \neg(F \vee G); \Sigma}{\Gamma, \neg F, \neg G; \Sigma}$$

$$(\rightarrow^+)\frac{\Gamma, F \rightarrow G; \Sigma}{\Gamma, \neg F; \Sigma \qquad \Gamma, G; \Sigma} \qquad (\rightarrow^-)\frac{\Gamma, \neg(F \rightarrow G); \Sigma}{\Gamma, F, \neg G; \Sigma} \qquad (\Box^-)\frac{\Gamma, \neg\Box\neg A; \Sigma}{A, \Box\neg A, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\sim^{\pm}}, \Sigma; \emptyset}$$

$$(\mid\sim^+)\frac{\Gamma, A \mid\sim B; \Sigma}{\Gamma, \neg A; \Sigma, A \mid\sim B \qquad \Gamma, \neg\Box\neg A; \Sigma, A \mid\sim B \qquad \Gamma, B; \Sigma, A \mid\sim B} \qquad (\mid\sim^-)\frac{\Gamma, \neg(A \mid\sim B); \Sigma}{A, \Box\neg A, \neg B, \Gamma^{\mid\sim^{\pm}}, \Sigma; \emptyset}$$

Figure 5.3: Tableau calculus $\mathcal{TP}^{\mathbf{T}}$.

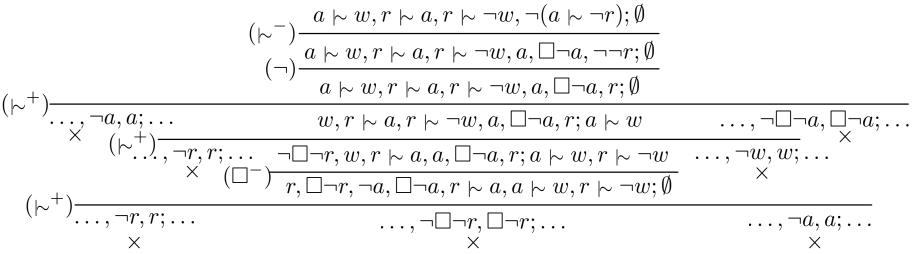

$(\mid\sim^-)$ $a \mid\sim w, r \mid\sim a, r \mid\sim \neg w, \neg(a \mid\sim \neg r); \emptyset$
$(\neg)$ $a \mid\sim w, r \mid\sim a, r \mid\sim \neg w, a, \Box\neg a, \neg\neg r; \emptyset$
$a \mid\sim w, r \mid\sim a, r \mid\sim \neg w, a, \Box\neg a, r; \emptyset$
$(\mid\sim^+)$
$\ldots, \neg a, a; \ldots$ ×
$w, r \mid\sim a, r \mid\sim \neg w, a, \Box\neg a, r; a \mid\sim w$
$\ldots, \neg\Box\neg a, \Box\neg a; \ldots$ ×
$(\mid\sim^+)$ $\ldots, \neg r, r; \ldots$ ×
$\neg\Box\neg r, w, r \mid\sim a, a, \Box\neg a, r; a \mid\sim w, r \mid\sim \neg w$
$\ldots, \neg w, w; \ldots$ ×
$(\Box^-)$ $r, \Box\neg r, \neg a, \Box\neg a, r \mid\sim a, a \mid\sim w, r \mid\sim \neg w; \emptyset$
$(\mid\sim^+)$
$\ldots, \neg r, r; \ldots$ ×
$\ldots, \neg\Box\neg r, \Box\neg r; \ldots$ ×
$\ldots, \neg a, a; \ldots$ ×

Figure 5.4: A derivation of *adult* $\mid\sim$ *worker*, *retired* $\mid\sim$ *adult*, *retired* $\mid\sim$ $\neg$*worker*, $\neg$(*adult* $\mid\sim$ $\neg$*retired*). For readability, we use a to denote *adult*, r for *retired*, and w for *worker*.

two applications are consecutive. Therefore, the second application of $(\mid\sim^+)$ is useless, since each of the conclusions has already been obtained after the first application, and can be removed.

■

Let us introduce a property of the tableau which will be crucial in many of the following proofs. Let us first define the notion of regular node.

Definition 5.15. *A node* $\Gamma; \Sigma$ *is called regular if the following condition holds:*

$$\textit{if } \neg\Box\neg A \in \Gamma, \textit{ then there is } A \mid\sim B \in \Gamma \cup \Sigma$$

It is easy to see that all nodes in a tableau starting from a pair $\Gamma; \emptyset$ are regular, when Γ is a set of formulas of $\mathcal{L}$. This is stated by the following proposition:

Proposition 5.16. *Given a pair* $\Gamma; \emptyset$*, where* Γ *is a set of formulas of* $\mathcal{L}$*, every tableau obtained by applying* $\mathcal{TP}^{\mathbf{T}}$*'s rules only contains regular nodes.*

Proof. Given a regular node $\Gamma; \Sigma$ and any rule of $\mathcal{TP}^{\mathbf{T}}$, we have to show that each conclusion of the rule is still a regular node. The proof is immediate for all the propositional rules, for $(\Box^-)$ and for $(\mid\sim^-)$, since no negated box formula not belonging to their premise is introduced in their conclusion(s). Consider now an application of $(\mid\!\sim^+)$ to a regular node $\Gamma', A \mid\!\sim B; \Sigma$: one of the conclusions is $\Gamma', \neg\Box\neg A; \Sigma, A \mid\!\sim B$, and it is a regular node since there is $A \mid\!\sim B$ in the auxiliary set of used conditional in correspondence of the negated boxed formula $\neg\Box\neg A$.

■

From now on, we can assume without loss of generality that only regular nodes may occur in a tableau. In order to prove that $\mathcal{TP}^{\mathbf{T}}$ ensures a terminating proof search, we define a complexity measure on a node $\Gamma; \Sigma$, denoted by $m(\Gamma; \Sigma)$, which consists of four measures c_1, c_2, c_3 and c_4 in a lexicographic order. We write $A \mid\!\sim B \in_+ \Gamma$ (resp. $A \mid\!\sim B \in_- \Gamma$) if $A \mid\!\sim B$ occurs positively (resp. negatively) in Γ, where positive and negative occurrences are defined in the standard way. We denote by $cp(F)$ the complexity of a formula F, defined as follows:

Definition 5.17 (Complexity of a formula)**.**

- $cp(P) = 1$, *where* $P \in ATM$
- $cp(\neg F) = 1 + cp(F)$
- $cp(F \bigotimes G) = 1 + cp(F) + cp(G)$, *where* $\bigotimes$ *is any binary boolean operator*
- $cp(\Box\neg A) = 1 + cp(\neg A)$
- $cp(A \mid\!\sim B) = 3 + cp(A) + cp(B)$.

Definition 5.18. *We define* $m(\Gamma; \Sigma) = \langle c_1, c_2, c_3, c_4 \rangle$ *where:*

- $c_1 =\mid \{A \mid\!\sim B \in_- \Gamma\} \mid$
- $c_2 =\mid \{A \mid\!\sim B \in_+ \Gamma \cup \Sigma \mid \Box\neg A \not\in \Gamma\} \mid$
- $c_3 =\mid \{A \mid\!\sim B \in_+ \Gamma\} \mid$
- $c_4 = \sum_{F \in \Gamma} cp(F)$

We consider the lexicographic order *given by* $m(\Gamma; \Sigma)$*, that is to say: given* $m(\Gamma; \Sigma) = \langle c_1, c_2, c_3, c_4 \rangle$ *and* $m(\Gamma'; \Sigma') = \langle c'_1, c'_2, c'_3, c'_4 \rangle$*, we say that* $m(\Gamma; \Sigma) < m(\Gamma'; \Sigma')$ *iff there exists* i*,* $i = 1, 2, 3, 4$*, such that the following conditions hold:*

- $c_i < c'_i$
- *for all* j*,* $0 < j < i$*, we have that* $c_j = c'_j$

Intuitively, c_1 is the number of negated conditionals to which the $(\mid\!\sim^-)$ rule can still be applied. An application of $(\mid\!\sim^-)$ reduces c_1. c_2 represents the number of positive conditionals *which can still create a new world*. The application of $(\Box^-)$ reduces c_2: indeed, if $(\mid\!\sim^+)$ is applied to $A \mid\!\sim B$, this application introduces a branch containing $\neg\Box\neg A$; when a new world is generated by an application of $(\Box^-)$ on $\neg\Box\neg A$, it contains A and $\Box\neg A$. If $(\mid\!\sim^+)$ is applied to $A \mid\!\sim B$ once again, then the conclusion where $\neg\Box\neg A$ is introduced leads to a closed branch, by the presence of $\Box\neg A$ in that branch. c_3 is the number of positive conditionals not yet considered in that branch. c_4 is the sum of the complexities of the formulas in Γ; an application of a boolean rule reduces c_4.

With the restriction on the application of the rules of Definition 5.13 at hand, we prove that the strategy does *not* generate any branch of infinite length. To this purpose we need the following lemma:

Lemma 5.19. *Let $\Gamma'; \Sigma'$ be obtained by an application of a rule of $\mathcal{TP}^{\mathbf{T}}$ to a premise $\Gamma; \Sigma$. Then, we have that either $m(\Gamma'; \Sigma') < m(\Gamma; \Sigma)$ or $\mathcal{TP}^{\mathbf{T}}$ leads to the construction of a closed tableau for $\Gamma'; \Sigma'$.*

Proof. We consider each rule of the calculus $\mathcal{TP}^{\mathbf{T}}$:

- $(\mid\!\sim^-)$: one can easily observe that the conditional formula $\neg(A \mid\!\sim B)$ to which this rule is applied does not belong to the only conclusion. Hence the measure c_1 in $m(\Gamma'; \Sigma')$, say $c_{1'}$, is smaller than c_1 in $m(\Gamma, \Delta)$, say c_1;
- $(\Box^-)$: no negated conditional is added nor deleted in the conclusions, thus $c_1 = c_1'$. Suppose we are considering an application of $(\Box^-)$ on a formula $\neg\Box\neg A$. We can observe the following facts:
  - the formula $\neg\Box\neg A$ has been introduced by an application of $(\mid\!\sim^+)$, being this one the only rule introducing a boxed formula in the conclusion; more precisely, it derives from an application of $(\mid\!\sim^+)$ on a conditional formula $A \mid\!\sim B$;
  - $A \mid\!\sim B$ belongs to both $\Gamma; \Sigma$ and $\Gamma'; \Sigma'$, since no rule of $\mathcal{TP}^{\mathbf{T}}$ removes positive conditionals (at most, the $(\mid\!\sim^+)$ rule *moves* conditionals from Γ to Σ);
  - $A \mid\!\sim B$ does not "contribute" to $c_{2'}$, since the application of $(\Box^-)$ introduces $\Box\neg A$ in the conclusion Γ' (remember that $c_{2'} = \mid \{A \mid\!\sim B \in_+ \Gamma' \cup \Sigma' \mid \Box\neg A \not\in \Gamma'\} \mid$).

  We distinguish two cases:

  1. $\Box\neg A$ does *not* belong to the premise of $(\Box^-)$: in this case, by the above facts, we can easily conclude that $c_{2'} < c_2$, since $\Box\neg A$ belongs only to the conclusion;
  2. $\Box\neg A$ belongs to the premise of $(\Box^-)$: we are considering a derivation of the following type:

  $$\frac{\Gamma, \Box\neg A, \neg\Box\neg A}{\Gamma^{\mid\sim\pm}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \neg A, A, \Box\neg A}\ (\Box^-)$$

     In this case, $c_{2'} = c_2$; however, we can conclude that the tableau built for $\Gamma^{\mid\sim\pm}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \neg A, A, \Box\neg A$ is closed, since:
     - A is a propositional formula
     - the restriction in Definition 5.13 leads to a proof in which the propositional rules and $(\mathbf{AX})$ are applied to A and $\neg A$ *before* $(\Box^-)$ is further applied. The resulting tableau is closed;

- $(\mid\!\sim^+)$: we have that $c_1 = c_1'$, since we have the same negated conditionals in the premise as in all the conclusions. The same for c_2, since the formula $A \mid\!\sim B$ to which the rule is applied is also maintained in the conclusions (it moves from unused to already used conditionals). We conclude that $m(\Gamma'; \Sigma') < m(\Gamma; \Sigma)$, since $c_{3'} < c_3$. Indeed, the $(\mid\!\sim^+)$ rule moves $A \mid\!\sim B$ from Γ to the set Σ of already considered conditionals, thus $A \mid\!\sim B \in \Gamma$ whereas $A \mid\!\sim B \not\in \Gamma'$;
- rules for the boolean connectives: it is easy to observe that c_1, c_2 and c_3 are the same in the premise and in any conclusion, since conditional formulas are side

formulas in the application of these rules. We conclude that $m(\Gamma'; \Sigma') < m(\Gamma; \Sigma)$ since $c_{4'} < c_4$. Indeed, the complexity of the formula to which the rule is applied is greater than (the sum of) the complexity of its subformula(s) introduced in the conclusion(s).

■

Now we have all the elements to prove the following theorem:

Theorem 5.20 (Termination of $\mathcal{T}\mathbf{P}^{\mathbf{T}}$)**.** $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ *ensures a terminating proof search.*

Proof. By Lemma 5.19 we know that in any open branch the value of $m(\Gamma; \Sigma)$ decreases each time a tableau rule is applied. Therefore, a finite number of applications of the rules leads either to build a closed tableau or to nodes $\Gamma; \Sigma$ such that $m(\Gamma; \Sigma)$ is *minimal*. In particular, we observe that, when the branch does not close, $m(\Gamma; \Sigma) = \langle 0, 0, 0, c_{4_{min}} \rangle$. In this case, no rule, with the exception of $(\Box^-)$, is applicable to $\Gamma; \Sigma$. Indeed, $(\mid\!\sim^-)$ rule is not applicable, since no negated conditional belongs to Γ (since $c_1 = 0$). If $(\Box^-)$ is applicable, then there is $\neg\Box\neg A \in \Gamma$, to which the rule is applied. However, since $c_2 = 0$, we have also that $\Box\neg A \in \Gamma$. Therefore, the conclusion of an application of $(\Box^-)$ contains both A and $\neg A$ and, by the restriction in Definition 5.13 and since A is propositional, the procedure terminates building a closed tableau. $(\mid\!\sim^+)$ is not applicable, since no positive conditionals belong to Γ (all positive conditionals $A \mid\!\sim B$ have been moved in Σ since $c_3 = 0$). Last, no rule for the boolean connectives is applicable, since c_4 assumes its minimal value $c_{4_{min}}$. For a contradiction, suppose one boolean rule is still applicable: by Lemma 5.19, the sum of the complexity of the formulas in the conclusion(s) decreases, i.e. c_4 in the conclusion(s) is smaller than in the premise $\Gamma; \Sigma$, against the minimality of this measure in $\Gamma; \Sigma$.

■

Optimal Proof Search Procedure and Complexity of P

We conclude this section with a complexity analysis of $\mathcal{T}\mathbf{P}^{\mathbf{T}}$, in order to define an optimal decision procedure for **P** based on our tableau calculus. Intuitively, we can obtain a **coNP** decision procedure, by taking advantage of the following facts:

1. Negated conditionals do not interact with the current world, nor they interact among themselves (by the disjunction property). Thus they can be handled separately and eliminated always as a first step.

2. We can replace the $(\Box^-)$ which is responsible of backtracking in the tableau construction by a stronger rule that does not need backtracking.

Regarding (1), by the disjunction property we can reformulate the $(\mid\!\sim^-)$ rule as follows:

$$\begin{array}{c} \Gamma, \neg(A \mid\!\sim B); \Sigma \\ \Sigma, A, \Box\neg A, \neg B, \Gamma^{\mid\sim^+}; \emptyset \end{array} \ (\mid\!\sim^-)$$

This rule reduces the length of a branch at the price of making the proof search more non-deterministic.

Regarding (2), we can adopt the following strengthened version of $(\Box^-)$. We use $\Gamma^{\Box^-}_{-i}$ to denote $\{\neg\Box\neg A_j \vee A_j \mid \neg\Box\neg A_j \in \Gamma \wedge j \neq i\}$.

$$\frac{\Gamma, \neg\Box\neg A_1, \neg\Box\neg A_2, ..., \neg\Box\neg A_n; \Sigma}{\Sigma, \Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, A_1, \Box\neg A_1, \Gamma^{\Box^{-}}_{-1}; \emptyset \mid \ldots \mid \Sigma, \Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, A_n, \Box\neg A_n, \Gamma^{\Box^{-}}_{-n}; \emptyset} \; (\Box_s^-)$$

Rule $(\Box_s^-)$ contains a branch for each $\neg\Box\neg A_i$ in Γ. On each branch, for a given $\neg\Box\neg A_i$, a new minimal world is created for A_i, where A_i and $\Box\neg A_i$ hold, and for all other $\neg\Box\neg A_j$, either A_j holds in that world or the formula $\neg\Box\neg A_j$ is recorded.
The advantage of this rule over the original $(\Box^-)$ rule is that no backtracking on the choice of the formula $\neg\Box\neg A_i$ is needed when searching for a closed tableau. The reason is that all alternatives are kept in the conclusion. As we will see below, by using this rule we can provide a tableau construction algorithm with no backtracking.

We call $L\mathcal{T}\mathbf{P^T}$ the calculus obtained by replacing in $\mathcal{T}\mathbf{P^T}$ the initial rules $(\mid\!\sim^-)$ and $(\Box^-)$ with the ones reformulated above. We can prove that $L\mathcal{T}\mathbf{P^T}$ is sound and complete with respect to the preferential models. To prove soundness, we consider the multi-linear models introduced in Section 3.2.2.

Theorem 5.21. *The rule* $(\Box_s^-)$ *is sound.*

Proof. Let $\Gamma = \Gamma', \neg\Box\neg A_1, \neg\Box\neg A_2, \ldots, \neg\Box\neg A_n$. We omit Σ for the sake of readability. We prove that if Γ is satisfiable then also one conclusion of the rule $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, A_i$, $\Box\neg A_i, \Gamma^{\Box^{-}}_{-i}$ is satisfiable. By Theorem 3.11, we can assume that Γ is satisfiable in a multi-linear model $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$ with $\mathcal{M}, x \models \Gamma$. Then there are $z_1 < x, \ldots, z_n < x$, such that $z_i \in Min_<(A_i)$; thus $\mathcal{M}, z_i \models A_i \wedge \Box\neg A_i$. We easily have also that $\mathcal{M}, z_i \models \Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}$. Since $\mathcal{M}$ is a multi-linear model, the z_i, $i = 1, 2, \ldots, n$, whenever distinct, are totally ordered: we have that $z_i < x$, so that they must belong to the same component. Let z_k be the maximum of z_i $(1 \leq k \leq n)$, i.e. for each z_i, $i = 1, 2, \ldots, n$, we have either (i) $z_i = z_k$ or (ii) $z_i < z_k$. In case (i), we have that $\mathcal{M}, z_k \models A_i$. In case (ii) we have that $\mathcal{M}, z_k \models \neg\Box\neg A_i$. We have shown that for each $i \neq k$, $\mathcal{M}, z_k \models A_i \vee \neg\Box\neg A_i$. We can conclude that $\mathcal{M}, z_k \models \Gamma^{\Box^{-}}_{-k}$. Thus $\mathcal{M}, z_k \models \Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, A_k, \Box\neg A_k, \Gamma^{\Box^{-}}_{-k}$, which is one of the conclusions of the rule.

■

We can prove that the calculus obtained by replacing the $(\Box^-)$ rule with its stronger version $(\Box_s^-)$ is complete with respect to the semantics:

Theorem 5.22. *The calculus* $L\mathcal{T}\mathbf{P^T}$ *is complete.*

Proof. We repeat the same construction as in the proof of Theorem 5.6, in order to build a preferential model, more precisely a multi-linear model, of a set of formulas Γ for which there is no closed tableau. As a difference with the construction in Theorem 5.6, we replace point 4. by the points 4_{strong}, $4a_{strong}$, $4b_{strong}$, and $4c_{strong}$, obtaining the following procedure (X is the set of worlds of the model):

```
1. initialize $X = \{\mathtt{SAT}(\Gamma; \Sigma)\}$; mark $\mathtt{SAT}(\Gamma; \Sigma)$ as unresolved;
**while** $X$ contains unresolved nodes **do**
    2. choose an unresolved $\Gamma_i; \Sigma_i$ from $X$;
    3. **for** each formula $\neg(A \mid\!\sim B) \in \Gamma_i$
        3a. let $(\Gamma; \Sigma)_{\neg(A \mid\!\sim B)}$ =$\mathtt{SAT}(\mathtt{APPLY}((\Gamma_i; \Sigma_i), \mid\!\sim^-, \neg(A \mid\!\sim B)))$;
        3b. **if** $(\Gamma; \Sigma)_{\neg(A \mid\!\sim B)} \notin X$ **then** $X = X \cup \{(\Gamma; \Sigma)_{\neg(A \mid\!\sim B)}\}$ and
                                        mark $(\Gamma; \Sigma)_{\neg(A \mid\!\sim B)}$ as unresolved;
    $4_{strong}$. if there is $\neg\Box\neg A \in \Gamma_i$ then
        $4a_{strong}$. let $(\Gamma_j;\Sigma_j)$ =SAT(APPLY($(\Gamma_i;\Sigma_i),\Box_s^-$));
        $4b_{strong}$. add the relation $(\Gamma_j;\Sigma_j) < (\Gamma_i;\Sigma_i)$;
        $4c_{strong}$. if $(\Gamma_j;\Sigma_j) \not\in X$ then $X = X \cup \{(\Gamma_j;\Sigma_j)\}$ and mark $(\Gamma_j;\Sigma_j)$ as unresolved;
    5. mark $\Gamma_i;\Sigma_i$ as resolved;
endWhile;
```

We denote by APPLY$((\Gamma;\Sigma),\Box_s^-)$ the result of applying the rule $(\Box_s^-)$ to $\Gamma;\Sigma$; as for the other branching rules, APPLY chooses one consistent conclusion of $(\Box_s^-)$ in an arbitrary but fixed manner. Facts 5.7 and 5.8 can be proved as in Theorem 5.6. This holds also for Fact 5.10 with one difference, for what concerns the case in which $\neg\Box\neg A \in \Gamma_i$. In this case, by construction there is a $\Gamma_j;\Sigma_j$ such that $(\Gamma_j;\Sigma_j) <_X (\Gamma_i;\Sigma_i)$. We can prove by induction on the length n of the chain $<_X$ starting from $\Gamma_i;\Sigma_i$ that if $\neg\Box\neg A \in \Gamma_i$, then $\mathcal{M},(\Gamma_i;\Sigma_i) \not\models \Box\neg A$. If $n = 1$, it must be the case that $A \in \Gamma_i$; hence, by inductive hypothesis on the structure of the formula, $\mathcal{M},(\Gamma_j;\Sigma_j) \models A$, thus $\mathcal{M},(\Gamma_i;\Sigma_i) \not\models \Box\neg A$. If $n > 1$, by the $(\Box_s^-)$ rule, either $A \in \Gamma_j$ and we conclude as in the previous case, or $\neg\Box\neg A \in \Gamma_j$ and the fact holds by inductive hypothesis on the length. From these facts, we can conclude that the model built is preferential and satisfies the initial set Γ, therefore $\mathcal{LT}\mathbf{P}^{\mathbf{T}}$ is complete.

■

In the following, we describe a rule application's strategy that allows us to decide the satisfiability of a node $\Gamma;\Sigma$ in **P** in non-deterministic polynomial time. As a first step, the strategy applies the boolean rules as long as possible. In case of branching rules, this operation nondeterministically selects (guesses) one of the conclusions of the rules. For each negated conditional, the strategy applies the rule $(\mid\!\sim^-)$ to it, generating a node $\Gamma';\Sigma'$ that does not contain any negated conditional[2]. On this node, the strategy applies the static rules as far as possible, then it iterates the following steps until either the current node contains an axiom or it does not contain negated boxed formulas $\neg\Box\neg A$:

1. apply the $(\Box_s^-)$ rule by guessing a branch;

2. apply the static rules as far as possible to the obtained node.

If the final node does not contain an axiom, then we conclude that $\Gamma';\Sigma'$ is satisfiable. If, for each application of $(\mid\!\sim^-)$, $\Gamma';\Sigma'$ is satisfiable, then the initial node $\Gamma;\Sigma$ is satisfiable.

The overall complexity of the strategy can be estimated as follows. Consider that $n =\mid \Gamma \cup \Sigma \mid$. The strategy builds a tableau branch for each of the $O(n)$ set of formulas obtained by applying $(\mid\!\sim^-)$ to each negated conditional (indeed, there are at most $O(n)$ negated conditionals). In order to do this, the strategy alternates applications of $(\Box_s^-)$ and of static rules (to saturate the obtained nodes). In case of branching rules, this saturation nondeterministically selects (guesses) one of the conclusions of the rules. The $(\Box_s^-)$ rule can be applied at most $O(n)$ times, since it can be applied only once for each formula $\neg\Box\neg A$, and the number of different negated box formulas is at most $O(n)$. Moreover, as the number of different subformulas of Γ is at most $O(n)$, in all steps involving the application of static rules, there at most

[2] By the new version of $(\mid\!\sim^-)$, we can consider negated conditionals one at a time. Indeed, for $\Gamma, \neg(A \mid\!\sim B), \neg(C \mid\!\sim D);\Sigma$ to be satisfiable, it is sufficient that both $\Gamma, \neg(A \mid\!\sim B);\Sigma$ and $\Gamma, \neg(C \mid\!\sim D);\Sigma$, separately considered, are satisfiable.

$O(n)$ applications of these rules. Therefore, the length of the tableau branch built by the strategy is $O(n^2)$. Finally, we can observe that all the nodes of the tableau contain a number of formulas which is polynomial in n, therefore to test that a node contains an axiom has at most complexity polynomial in n.

The above strategy allows the satisfiability of a set of formulas of logic **P** to be decided in nondeterministic polynomial time. Given that the problem of deciding validity for preferential logic **P** is known to be **coNP**-complete [KLM90], we can conclude that the above strategy is optimal for **P**.

Theorem 5.23 (Complexity of **P**)**.** *The problem of deciding validity for* **P** *is* **coNP***-complete.*

5.3 A Tableau Calculus for Loop Cumulative Logic CL

In this section we develop a tableau calculus $\mathcal{T}$**CL** for **CL**, and we show that it can be turned into a terminating calculus. This provides a decision procedure for **CL** and a **coNP**-membership upper bound for validity in **CL**.

The calculus $\mathcal{T}$**CL** can be obtained from the calculus $\mathcal{T}$**P** for preferential logics, by adding a suitable rule (L^-) for dealing with the modality L introduced in Section 3.2.3. As already mentioned in Section 3.2.3, the formulas that appear in the tableaux for **CL** belong to the language $\mathcal{L}_L$ obtained from $\mathcal{L}$ as follows: (i) if A is propositional, then $A \in \mathcal{L}_L$; $LA \in \mathcal{L}_L$; $\Box\neg LA \in \mathcal{L}_L$; (ii) if A, B are propositional, then $A \mid\!\sim B \in \mathcal{L}_L$; (iii) if F is a boolean combination of formulas of $\mathcal{L}_L$, then $F \in \mathcal{L}_L$. Observe that the only allowed combination of $\Box$ and L is in formulas of the form $\Box\neg LA$ where A is propositional.

We define:

$$\Gamma^{L^{\downarrow}} = \{A \mid LA \in \Gamma\}$$

Our tableau system $\mathcal{T}$**CL** is shown in Figure 5.5. Observe that rules $(\mid\!\sim^+)$ and $(\mid\!\sim^-)$ have been changed as they introduce the modality L in front of the propositional formulas A and B in their conclusions. This straightforwardly corresponds to the semantics of conditionals in CL preferential models (see Definition 3.13). The new rule (L^-) is a dynamic rule.

Let us now prove that $\mathcal{T}$**CL** is sound and complete with respect to loop-cumulative models.

Theorem 5.24 (Soundness of $\mathcal{T}$**CL**)**.** *The system* $\mathcal{T}$**CL** *is sound with respect to CL-preferential models, i.e. given a set of formulas* Γ*, if there is a closed tableau* Γ*, then* Γ *is unsatisfiable.*

Proof. We show that for all the rules in $\mathcal{T}$**CL**, if the premise is satisfiable by a CL-preferential model then also one of the conclusions is. As far as the rules already present in $\mathcal{T}$**P** are concerned, the proof is very similar, with the only exception that we have to substitute A in the proof by LA.

Let us now consider the new rule (L^-). Let $\mathcal{M}, w \models \Gamma, \neg LA$ where $\mathcal{M} = \langle \mathcal{W}, R, <, V\rangle$ is a CL-preferential model and $w \in \mathcal{W}$. Then there is $w' \in \mathcal{W}$ s.t. $(w, w') \in R$ and $\mathcal{M}, w' \models \neg A$. Furthermore, $\mathcal{M}, w' \models \Gamma^{L^{\downarrow}}$. It follows that the conclusion of the

$$(\mathrel{|\!\sim}^+)\ \frac{\Gamma, A \mathrel{|\!\sim} B}{\Gamma, A \mathrel{|\!\sim} B, \neg LA \qquad \Gamma, A \mathrel{|\!\sim} B, \neg\Box\neg LA \qquad \Gamma, A \mathrel{|\!\sim} B, LB}$$

$$(\mathrel{|\!\sim}^-)\ \frac{\Gamma, \neg(A \mathrel{|\!\sim} B)}{\Gamma^{\mathrel{|\!\sim}^{\pm}}, LA, \Box\neg LA, \neg LB} \qquad (\Box^-)\ \frac{\Gamma, \neg\Box\neg LA}{\Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mathrel{|\!\sim}^{\pm}}, LA, \Box\neg LA}$$

$$(L^-)\ \frac{\Gamma, \neg LA}{\Gamma^{L^{\downarrow}}, \neg A}\ ;\ \frac{\Gamma}{\Gamma^{L^{\downarrow}}}\ \text{if } \Gamma \text{ does not contain negated } L\text{-formulas} \qquad (\mathbf{AX})\ \Gamma, P, \neg P \quad \text{with } P \in ATM$$

Figure 5.5: Tableau system $\mathcal{T}\mathbf{CL}$. If there are no negated L-formulas $\neg LA$ in the premise of (L^-), then the rule allows to step from Γ to $\Gamma^{L^{\downarrow}}$. To save space, the boolean rules are omitted.

rule is satisfiable. If Γ does not contain negated L-formulas, since R is serial, we still have that there exists $w' \in \mathcal{W}$ s.t. $(w, w') \in R$, and $\mathcal{M}, w' \models \Gamma^{L^{\downarrow}}$. Hence the conclusion is satisfiable.

■

Soundness with respect to loop-cumulative models in Definition 3.12 follows from the correspondence established by Proposition 3.14.

The proof of the completeness of the calculus can be done as for the preferential case, provided we suitably modify the procedure for constructing a model for a finite consistent set of formulas Γ of $\mathcal{L}_L$. First of all, we modify the definition of saturated sets as follows:

- if $A \mathrel{|\!\sim} B \in \Gamma$ then $\neg LA \in \Gamma$ or $\neg\Box\neg LA \in \Gamma$ or $LB \in \Gamma$

For this notion of saturated set of formulas we can still prove Lemma 5.5 for language $\mathcal{L}_L$.

Theorem 5.25 (Completeness of $\mathcal{T}\mathbf{CL}$). *$\mathcal{T}\mathbf{CL}$ is complete with respect to CL-preferential models, i.e. if a set of formulas Γ is unsatisfiable, then Γ has a closed tableau in $\mathcal{T}\mathbf{CL}$.*

Proof. We define a procedure for constructing a model satisfying a consistent set of formulas $\Gamma \in \mathcal{L}_L$ by modifying the procedure for the preferential logic **P**. We add to the procedure used in the proof of Theorem 5.6 the new steps 4' and 4" between step 4 and step 5, obtaining the following procedure:

```
1. initialize $X = \{\mathtt{SAT}(\Gamma)\}$; mark $\mathtt{SAT}(\Gamma)$ as unresolved;
while $X$ contains unresolved nodes do
    2. choose an unresolved $\Gamma_i$ from $X$;
    3. for each formula $\neg(A \mathrel{|\!\sim} B) \in \Gamma_i$
        3a. let $\Gamma_{\neg(A \mathrel{|\!\sim} B)} = \mathtt{SAT}(\mathtt{APPLY}(\Gamma_i, \mathrel{|\!\sim}^-, \neg(A \mathrel{|\!\sim} B)))$;
        3b. if $\Gamma_{\neg(A \mathrel{|\!\sim} B)} \notin X$ then $X = X \cup \{\Gamma_{\neg(A \mathrel{|\!\sim} B)}\}$ and mark $\Gamma_{\neg(A \mathrel{|\!\sim} B)}$ as unresolved;
    4. for each formula $\neg\Box\neg LA \in \Gamma_i$, let $(\Gamma;\Sigma)_{\neg\Box\neg LA} = \mathtt{SAT}(\mathtt{APPLY}(\Gamma_i, \Box^-, \neg\Box\neg LA))$;
        4a. add the relation $\Gamma_{\neg\Box\neg LA} < \Gamma_i$;
        4b. if $\Gamma_{\neg\Box\neg LA} \notin X$ then $X = X \cup \{\Gamma_{\neg\Box\neg LA}\}$ and mark $\Gamma_{\neg\Box\neg LA}$ as unresolved;
    4'. if $\{\neg LA \mid \neg LA \in \Gamma_i\} \neq \emptyset$ then
            for each $\neg LA \in \Gamma_i$, let $\Gamma_{\neg LA})$ =SAT(APPLY($\Gamma_i, L^-, \neg LA$));
                4' a. add $(\Gamma_i, \Gamma_{\neg LA})$ to $R$;
                4' b. if $\Gamma_{\neg LA} \notin X$ then $X = X \cup \{\Gamma_{\neg LA}\}$ and mark $\Gamma_{\neg LA}$ as unresolved;
    4". else if $\Gamma_i^{L^\downarrow} \neq \emptyset$ then let $\Gamma_j$ =SAT(APPLY($\Gamma_i, L^-$));
            4" a. add $(\Gamma_i, \Gamma_j)$ to $R$;
            4" b. if $\Gamma_j \notin X$ then $X = X \cup \{\Gamma_j\}$ and mark $\Gamma_j$ as unresolved;
    5. mark $\Gamma_i$ as resolved;
endWhile;
```

We denote by APPLY(Γ, L^-) the result of applying the rule (L^-) to Γ, in case Γ does not contain any negated L-formula $\neg LA$. The procedure above terminates. The argument is similar to the case of **P**. In addition, observe that the worlds introduced by an application of (L^-) cannot lead to generate any further world by means of a dynamic rule, as they do not contain any boxed formula or L-formula.

We construct the model $\mathcal{M} = \langle X, R_X, <_X, V\rangle$ by defining X and V as in the case of **P**. We then define R_X as the least relation including all pairs in R augmented with the pairs satisfying the following conditions:

(i) if $\Gamma_i \in X$ and Γ_i has no R-successor, then $(\Gamma_i, \Gamma_i) \in R_X$;

(ii) if $(\Gamma_k, \Gamma_i) \in R_X$ and $(\Gamma_k, \Gamma_j) \in R_X$, then
$(\Gamma_i, \Gamma_j) \in R_X$.

Notice that condition (ii) is the euclidean closure of R. Last, we define $<_X$ as follows:

(iii) if $\Gamma_j < \Gamma_i$, then $\Gamma_j <_X \Gamma_i$;

(iv) if $\Gamma_j < \Gamma_i$, and $\Gamma_i R_X \Gamma_k$, then $\Gamma_j <_X \Gamma_k$;

(v) if $\Gamma_j <_X \Gamma_i$ and $\Gamma_i <_X \Gamma_k$, then $\Gamma_j <_X \Gamma_k$, i.e. $<_X$ is transitive.

Notice that the above conditions on R_X and $<_X$ are needed since the procedure builds two different kinds of worlds:

- *bad* worlds, obtained by an application of (L^-);
- *good* worlds: the other ones.

Bad worlds are those obtained by an application of (L^-). These worlds "forget" the positive conditionals in the initial set of formulas; for instance, if a world $\Gamma_i = \{\neg C, D\}$ is a bad world obtained from $\Gamma_j = \{A \mid\!\sim B, \neg LC, LD\}$, then it is "incomplete" by the absence of $A \mid\!\sim B$. The above supplementary conditions on R_X and $<_X$ are needed in order to prove that, even in presence of bad worlds, we can build a CL-preferential model satisfying the initial set of formulas, as shown below by Fact 5.26. This is not the only way to proceed: as an alternative, one should define R_X as obtained from R only by adding (Γ, Γ) for each Γ having no R-successors (in order to build a serial accessibility relation), then describe a saturation process for the bad worlds.

It is easy to show that the following properties hold for $\mathcal{M}$:

Fact 5.26. *For all* $\Gamma_i, \Gamma_j \in X$, *if* $(\Gamma_i, \Gamma_j) \in R_X$ *and* $LA \in \Gamma_i$ *then* $A \in \Gamma_j$.

Proof of Fact 5.26. Suppose that $(\Gamma_i, \Gamma_j) \in R_X$ and $LA \in \Gamma_i$. We distinguish two cases. First, we consider the case in which $\Gamma_i \neq \Gamma_j$. In this case, $(\Gamma_i, \Gamma_j) \in R$ and it has been added to R by step 4' or step 4" of the procedure above. Indeed, for all (Γ_i, Γ_j) that have been introduced because $(\Gamma_k, \Gamma_i) \in R$ and $(\Gamma_k, \Gamma_j) \in R$, both Γ_i and Γ_j derive from the application of (L^-) to Γ_k, hence they only contain propositional formulas and do not contain any LA. Hence, we have two different subcases:

- the relation (Γ_i, Γ_j) has been added to R by step 4': in this case, we have that $\neg LB \in \Gamma_i$ for some B. We can conclude that $A \in \Gamma_j$ by construction, since for each $LA \in \Gamma_i$ we have that $A \in \Gamma_j$ as a result of the application of $\texttt{SAT(APPLY((}\Gamma_i, L^-, \neg LB\texttt{))}$;
- the relation (Γ_i, Γ_j) has been added to R by step 4": similarly to the previous case, for each $LA \in \Gamma_i$, we have that A is added to Γ_j by construction.

Second, we consider the case in which $\Gamma_i = \Gamma_j$. In this case, it must be that (Γ_i, Γ_i) has been added to R_X, as Γ_i has no R-successors. This means that Γ_i does not contain formulas of the form LA or $\neg LA$, otherwise it would have an R-successor.

□ *Fact 5.26*

Fact 5.27. *For all formulas F and for all worlds $\Gamma_i \in X$ we have that: (i) if $F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \models F$; (ii) if $\neg F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \not\models F$.*

Proof of Fact 5.27. The proof is similar to the one for the preferential case. If F is an atom or a boolean combination of formulas, the proof is easy and left to the reader. We consider the other cases:

- $LA \in \Gamma_i$: we have to show that $\mathcal{M}, \Gamma_i \models LA$. Let Γ_j such that $(\Gamma_i, \Gamma_j) \in R_X$. By Fact 5.26, we conclude $A \in \Gamma_j$. By inductive hypothesis, then $\mathcal{M}, \Gamma_j \models A$, and we are done.
- $\neg LA \in \Gamma_i$: we have to show that $\mathcal{M}, \Gamma_i \not\models LA$. By construction (step $4'$ in the procedure) there must be a $\Gamma_j \in X$ such that $\neg A \in \Gamma_j$. By inductive hypothesis, $\mathcal{M}, \Gamma_j \not\models A$, which concludes the proof.
- $\Box\neg LA \in \Gamma_i$. For all $\Gamma_j <_X \Gamma_i$ we have $\neg LA \in \Gamma_j$ by the definition of $(\Box^-)$, since Γ_j has been generated by a sequence of applications of $(\Box^-)$ (notice that point (iv) in the definition of $<_X$ above does not play any role here, since this point only concerns nodes Γ_i that do not contain boxed or negated box formulas). By inductive hypothesis $\mathcal{M}, \Gamma_j \not\models LA$ for all $\Gamma_j <_X \Gamma_i$, whence $\mathcal{M}, \Gamma_i \models \Box\neg LA$.
- $\neg\Box\neg LA \in \Gamma_i$. By construction there is a Γ_j s.t. $\Gamma_j <_X \Gamma_i$ and $LA \in \Gamma_j$. By inductive hypothesis $\mathcal{M}, \Gamma_j \models LA$. Thus, $\mathcal{M}, \Gamma_i \not\models \Box\neg LA$.
- $A \mid\!\sim B \in \Gamma_i$. Let $\Gamma_j \in Min_{<_X}(LA)$. We distinguish two cases:
  - $A \mid\!\sim B \in \Gamma_j$, one can observe that $(1)\neg LA \in \Gamma_j$ or $(2)\neg\Box\neg LA \in \Gamma_j$ or $(3)LB \in \Gamma_j$, since Γ_j is saturated. (1) cannot be the case, since by inductive hypothesis $\mathcal{M}, \Gamma_j \not\models LA$, which contradicts the definition of $Min_{<_X}(LA)$. If (2), by construction of $\mathcal{M}$ there exists a set $\Gamma_k <_X \Gamma_j$ such that $LA \in \Gamma_k$. By inductive hypothesis $\mathcal{M}, \Gamma_k \models LA$, which contradicts $\Gamma_j \in Min_{<_X}(LA)$. Therefore, it must be that $(3)LB \in \Gamma_j$, and by inductive hypothesis $\mathcal{M}, \Gamma_j \models LB$. We conclude that $\mathcal{M}, \Gamma_i \models A \mid\!\sim B$.

  - $A \mid\!\sim B \not\in \Gamma_j$. Since all the rules apart from (L^-) preserve the conditionals, Γ_j must have been generated by applying (L^-) to some $\Gamma_k \in X$, i.e. Γ_j is a *bad world*. Hence, $(\Gamma_k, \Gamma_j) \in R_X$. In turn, it can be easily shown that Γ_k itself cannot have been generated by (L^-), hence $A \mid\!\sim B \in \Gamma_k$ and, since Γ_k is saturated, either (1)$\neg LA \in \Gamma_k$ or (2)$\neg\Box\neg LA \in \Gamma_k$ or (3)$LB \in \Gamma_k$. (1) is not possible, since by inductive hypothesis, it would entail that $\mathcal{M}, \Gamma_k \not\models LA$, i.e. there is Γ_l such that $(\Gamma_k, \Gamma_l) \in R_X$ and $\mathcal{M}, \Gamma_l \not\models A$. By point (ii) in the definition of R_X above, also $(\Gamma_j, \Gamma_l) \in R_X$, hence also $\mathcal{M}, \Gamma_j \not\models LA$, which contradicts $\Gamma_j \in Min_{<_X}(LA)$. If (2), by construction of $\mathcal{M}$ there exists a node $\Gamma_l <_X \Gamma_k$ such that $LA \in \Gamma_l$. By point (iv) in the definition of $<_X$ above, $\Gamma_l <_X \Gamma_j$, which contradicts $\Gamma_j \in Min_{<_X}(LA)$, since by inductive hypothesis $\mathcal{M}, \Gamma_l \models LA$. It follows that $LB \in \Gamma_k$. By inductive hypothesis $\mathcal{M}, \Gamma_k \models LB$, hence also $\mathcal{M}, \Gamma_j \models LB$. Indeed, since Γ_j does not contain any L-formula, by construction of the model and by point (ii) in the definition of R_X above, $(\Gamma_j, \Gamma_l) \in R_X$ just in case $(\Gamma_k, \Gamma_l) \in R_X$, from which the result follows. Hence, we can conclude $\mathcal{M}, \Gamma_i \models A \mid\!\sim B$.

- $\neg(A \mid\!\sim B) \in \Gamma_i$: by construction of X, there exists $\Gamma_j \in X$ such that $LA, \Box\neg LA, \neg LB \in \Gamma_j$. By inductive hypothesis we have that $\mathcal{M}, \Gamma_j \models LA$ and $\mathcal{M}, \Gamma_j \models \Box\neg LA$. It follows that $\Gamma_j \in Min_{<_X}(LA)$. Furthermore, always by induction, $\mathcal{M}, \Gamma_j \not\models LB$. Hence, $\mathcal{M}, \Gamma_i \not\models A \mid\!\sim B$.

□ *Fact 5.27*

Similarly to the case of **P**, it is easy to prove the following Fact:

Fact 5.28. *The relation $<_X$ is irreflexive, transitive, and satisfies the smoothness condition.*

Moreover:

Fact 5.29. *The relation R_X is serial.*

From the above Facts, we can conclude that $\mathcal{M} = \langle X, R_X, <_X, V\rangle$ is a CL-preferential model satisfying Γ, which concludes the proof of completeness.

■

From the above Theorem 5.25, together with Proposition 3.14, it follows that for any boolean combination of conditionals Γ, if there is no closed tableau for Γ, then Γ is satisfiable in a loop-cumulative model.

Similarly to the case of **P**, the above construction allows us to prove the following corollary:

Corollary 5.30 (Finite model property)**. CL** *has the finite model property.*

$$(\mid\!\sim^{+})\,\frac{\Gamma, A \mid\!\sim B; \Sigma}{\Gamma, \neg LA; \Sigma, A \mid\!\sim B \qquad \Gamma, \neg\Box\neg LA; \Sigma, A \mid\!\sim B \qquad \Gamma, LB; \Sigma, A \mid\!\sim B} \qquad (\mid\!\sim^{-})\,\frac{\Gamma, \neg(A \mid\!\sim B); \Sigma}{LA, \Box\neg LA, \neg LB, \Gamma^{\mid\sim^{\pm}}, \Sigma; \emptyset}$$

$$(\Box^{-})\,\frac{\Gamma, \neg\Box\neg LA; \Sigma}{LA, \Box\neg LA, \Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\sim^{\pm}}, \Sigma; \emptyset} \qquad (L^{-})\,\frac{\Gamma, \neg LA; \Sigma}{\Gamma^{L^{\downarrow}}, \neg A; \emptyset}\,;\ \frac{\Gamma; \Sigma}{\Gamma^{L^{\downarrow}}; \emptyset}\ \text{if } \Gamma \text{ does not contain negated } L-\text{formulas}$$

Figure 5.6: Tableau calculus $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$.

5.3.1 Decidability and Complexity of CL

Let us now analyze the calculus $\mathcal{T}\mathbf{CL}$ in order to obtain a decision procedure for **CL** logic. First of all, we reformulate the calculus as we have done for **P**, obtaining a system called $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$: we reformulate the $(\mid\!\sim^{+})$ rule so that it applies only once to each conditional in each world, by adding an extra set Σ. We reformulate the other rules accordingly, as shown in Figure 5.6. Moreover, we adopt the same restriction on the order of application of the rules in Definition 5.13.

As an example, we use $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ in order to show that $A \vee A \mid\!\sim B$ can be inferred from $A \mid\!\sim B \wedge C$. To this aim, a closed tableau for $A \mid\!\sim B \wedge C, \neg(A \vee A \mid\!\sim B); \emptyset$ is presented in Figure 5.7.

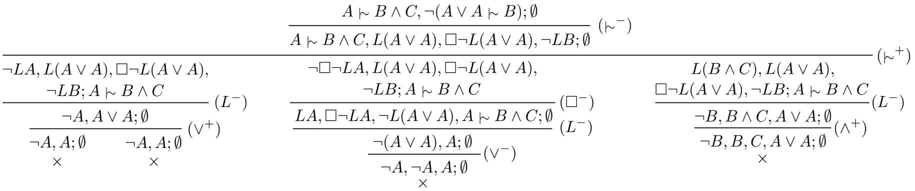

$A \mid\!\sim B \wedge C, \neg(A \vee A \mid\!\sim B); \emptyset$
$(\mid\!\sim^{-})$
$A \mid\!\sim B \wedge C, L(A \vee A), \Box\neg L(A \vee A), \neg LB; \emptyset$
$(\mid\!\sim^{+})$
$\neg LA, L(A \vee A), \Box\neg L(A \vee A), \neg LB; A \mid\!\sim B \wedge C$
(L^{-})
$\neg A, A \vee A; \emptyset$
$(\vee^{+})$
$\neg A, A; \emptyset$
×
$\neg A, A; \emptyset$
×
$\neg\Box\neg LA, L(A \vee A), \Box\neg L(A \vee A), \neg LB; A \mid\!\sim B \wedge C$
$(\Box^{-})$
$LA, \Box\neg LA, \neg L(A \vee A), A \mid\!\sim B \wedge C; \emptyset$
(L^{-})
$\neg(A \vee A), A; \emptyset$
$(\vee^{-})$
$\neg A, \neg A, A; \emptyset$
×
$L(B \wedge C), L(A \vee A), \Box\neg L(A \vee A), \neg LB; A \mid\!\sim B \wedge C$
(L^{-})
$\neg B, B \wedge C, A \vee A; \emptyset$
$(\wedge^{+})$
$\neg B, B, C, A \vee A; \emptyset$
×

Figure 5.7: A derivation in $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ of $A \mid\!\sim B \wedge C, \neg(A \vee A \mid\!\sim B); \emptyset$.

As far as termination is concerned, notice that the rule (L^{-}) can only be applied a finite number of times. Indeed, if it is applied to a premise $\Gamma, \neg LA$, then the conclusion only contains propositional formulas $\Gamma^{L^{\downarrow}}, \neg A$, and the rule (L^{-}) is no further applicable. The same holds in the case (L^{-}) is applied to a premise Γ such that Γ only contains positive L-formulas of the form LA (and does not contain negated L-formulas of the form $\neg LA$). Notice that (L^{-}) can also be applied to a node *not containing* any formula LA or $\neg LA$: in this case, the conclusion of the rule corresponds to an empty node $\emptyset$. Therefore, we reformulate (L^{-}) only by adding the extra set Σ of conditionals; the reformulated rule is shown in Figure 5.8.
Exactly as we made for **P**, we consider a lexicographic order given by $m(\Gamma; \Sigma) = \langle c_1, c_2, c_3, c_4 \rangle$ (Definition 5.18), and easily prove that each application of the rules of

$$(L^{-})\ \frac{\Gamma, \neg LA; \Sigma}{\Gamma^{L^{\downarrow}}, \neg A; \emptyset}\ ;\ \frac{\Gamma; \Sigma}{\Gamma^{L^{\downarrow}}; \emptyset} \quad \text{if } \Gamma \text{ does not contain negated } L-\text{formulas}$$

Figure 5.8: The rule (L^{-}) reformulated in $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$.

$\mathcal{TCL}^{\mathbf{T}}$ reduces this measure, as stated by the following Lemma:

Lemma 5.31. *Consider an application of any rule of* $\mathcal{TCL}^{\mathbf{T}}$ *to a premise* $\Gamma; \Sigma$ *and be* $\Gamma'; \Sigma'$ *any conclusion obtained; we have that either* $m(\Gamma'; \Sigma') < m(\Gamma; \Sigma)$ *or* $\mathcal{TCL}^{\mathbf{T}}$ *leads to the construction of a closed tableau for* $\Gamma'; \Sigma'$.

Proof. Identical to the proof of Lemma 5.19. Just observe that if (L^-) is applied, then c_1, c_2, and c_3 become 0, since conditional formulas are not kept in the conclusion. If the premise contains only L-formulas, then c_1, c_2 and c_3 are already equal to 0 in both the premise and the conclusion, but c_4 decreases, since (at least) one formula $(\neg)LA$ in the premise is removed, and a formula with a lower complexity ($\neg A$ or A) is introduced in the conclusion.

■

Furthermore, the decision algorithm for **P** described in Section 5.2.1 can be adapted to **CL**. To this aim, we observe that the disjunction property holds for **CL**, and this allows us to change the rule for negated conditionals in order to treat them independently as we have done for **P**. Moreover, we can replace the $(\Box^-)$ rule by a stronger rule that does not require backtracking in the tableau construction. The rule is the following ($\Gamma^{\Box^-}_{-i}$ is used to denote $\{\neg\Box\neg LA_j \vee LA_j \mid \neg\Box\neg LA_j \in \Gamma \wedge j \neq i\}$):

$$\frac{\Gamma, \neg\Box\neg LA_1, \neg\Box\neg LA_2, ..., \neg\Box\neg LA_n; \Sigma}{\Sigma, \Gamma^{\mid\sim\pm}, \Gamma^{\Box}, \Gamma^{\Box^\downarrow}, LA_1, \Box\neg LA_1, \Gamma^{\Box^-}_{-1}; \emptyset \mid \ldots \mid \Sigma, \Gamma^{\mid\sim\pm}, \Gamma^{\Box}, \Gamma^{\Box^\downarrow}, LA_n, \Box\neg LA_n, \Gamma^{\Box^-}_{-n}; \emptyset} \; (\Box^-_s)$$

By reasoning similarly to what done for **P**, we can show that the calculus in which $(\Box^-)$ is replaced by $(\Box^-_s)$ is sound and complete with respect to multi-linear CL-preferential models introduced in Definition 3.25. We get a decision procedure as in the case of **P** by defining a rule application's strategy that allows us to decide the satisfiability of a set of formulas in **CL** in non-deterministic polynomial time. As for **P**, the strategy checks the satisfiability of the nodes $\Gamma; \Sigma$ obtained by the application of the $(\mid\sim^-)$ rule to each negated conditional. In each case, we repeatedly apply $(\Box^-_s)$ for each negated boxed formula in Γ. At each step of application of $(\mid\sim^-)$ and $(\Box^-_s)$ rules, the obtained nodes is saturated with respect to the static rules as well as with respect to the (L^-) rule. In case of branching rules, this saturation nondeterministically selects (guesses) one of the conclusions of the rules.

As there is no nesting of box formulas within L modalities, the number of applications of the $(\Box^-_s)$ rule (for each application of $(\mid\sim^-)$) is polynomial in n, as in the case of **P**. As **coNP**-hardness immediately follows from the fact that **CL** includes classical logic, we obtain the following result:

Theorem 5.32 (Complexity of **CL**). *The problem of deciding validity for* **CL** *is* **coNP**-*complete.*

5.4 A Tableau Calculus for Cumulative Logic C

As for **CL**, we present a tableau calculus $\mathcal{TC}$ for cumulative logic **C** based on the mapping between Cumulative models and C-Preferential models. By the lack of transitivity of the preference relation, the smoothness condition can no longer be identified with the finite-chain condition of logic G. The interplay between rules ($\mid\!\sim^+$) and ($\Box^-$) may lead to generate infinite branches. Hence, termination cannot be ensured by preventing multiple applications of ($\mid\!\sim^+$) on the same conditional formula. In this case, we also need a loop-checking mechanism to prevent repeated expansions of the same tableau node. Nodes in $\mathcal{T}\mathbf{C}$ are *sets of formulas* Γ, rather than pairs $\Gamma; \Sigma$, since we do not need to keep track of positive conditionals already expanded in a branch.

In order to provide a calculus for $\mathbf{C}$, we have to replace the rule ($\Box^-$) with the weaker ($\Box^{\mathbf{C}-}$):

$$(\Box^{\mathbf{C}-})\frac{\Gamma, \neg\Box\neg LA}{\Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\sim^{\pm}}, LA}$$

Observe that, if we ignore conditionals, this rule is nothing else than the standard rule of modal logic K. This rule is weaker than the corresponding rule of the two other systems in two respects: (i) transitivity is not assumed thus we no longer have $\Gamma^{\Box}$ in the conclusion; (ii) the smoothness condition does no longer ensure that if $\neg\Box\neg LA$ is true in one world, then there is a smaller *minimal* world satisfying LA. This only happens if the world satisfies LA; thus $\Box\neg LA$ is dropped from the conclusion as well.

Moreover, we add the following form of the cut rule:

$$(\text{Weak-Cut})\frac{\Gamma}{\Gamma, \neg LA \qquad \Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad \Gamma, \Box\neg LA}$$

Intuitively, this rule takes care of enforcing the smoothness condition, and it can be applied to all L-formulas.

The (Weak-Cut) rule is not eliminable, as shown by the following example[3]. Let $\Gamma = \{\neg(A \mid\!\sim C), A \mid\!\sim B, B \mid\!\sim A, B \mid\!\sim C\}$, which is unsatisfiable in $\mathbf{C}$. Γ has a closed tableau only if we use (Weak-Cut) in the calculus, as shown by the derivation in Figure 5.9. Without (Weak-Cut), the above set of formulas does not have any closed tableau.

By incorporating the (Weak-Cut) rule we obtain a non-analytic calculus. Fortunately, we will show that the rule can be restricted so that it only applies to formulas LA such that A is the antecedent of a positive conditional formula in Γ, thus making the resulting calculus analytic. In order to prove this, we simplify the calculus by incorporating the application of ($\Box^{\mathbf{C}-}$) and the restricted form of (Weak-Cut) in the ($\mid\!\sim^+$) rule. The resulting calculus, called $\mathcal{T}\mathbf{C}$ and given in Figure 5.10, is equivalent to the calculus that would be obtained from the calculus $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ by replacing ($\Box^-$) with ($\Box^{\mathbf{C}-}$) and by introducing (Weak-Cut) restricted to antecedents of positive conditionals. The advantage of the adopted formulation is not only that it is more compact, but also that it allows a simpler proof of the admissibility of the non-restricted (Weak-Cut) (see Theorem 5.40 below).

Observe that the calculus does not contain any rule for negated box formulas, as the modified ($\mid\!\sim^+$) rule does no longer introduce formulas of the form $\neg\Box\neg LA$. The resulting language $\mathcal{L}_{L'}$ of the formulas appearing in a tableau for $\mathbf{C}$ extends $\mathcal{L}$ by formulas LA and $\Box\neg LA$, where A is propositional. Notice that negated boxed formulas $\neg\Box\neg LA$ do not appear in $\mathcal{L}_{L'}$, which is therefore a restriction of $\mathcal{L}_L$ used in $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$.

[3] Interestingly enough, this set of formulas corresponds to an instance of the well known (CSO) axiom of conditional logics $(A \Rightarrow B) \wedge (B \Rightarrow A) \wedge (A \Rightarrow C) \rightarrow (B \Rightarrow C)$.

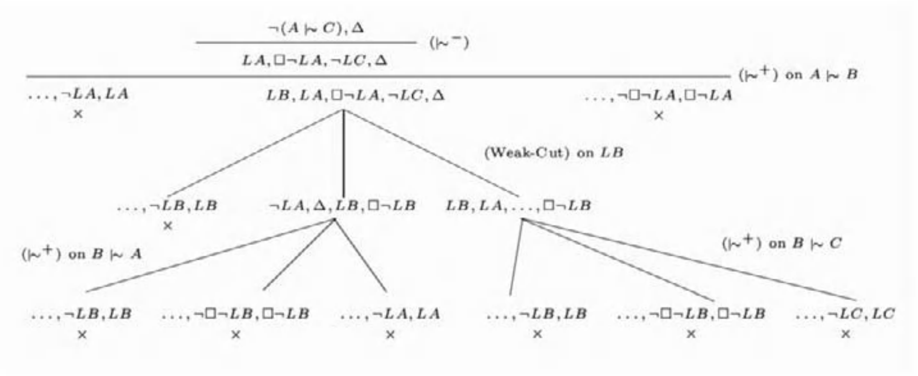

$\neg(A \mid\sim C), \Delta$
$(\mid\sim^-)$
$LA, \Box\neg LA, \neg LC, \Delta$
$(\mid\sim^+)$ on $A \mid\sim B$
$\dots, \neg LA, LA$
×
$LB, LA, \Box\neg LA, \neg LC, \Delta$
$\dots, \neg\Box\neg LA, \Box\neg LA$
×
(Weak-Cut) on LB
$\dots, \neg LB, LB$
×
$\neg LA, \Delta, LB, \Box\neg LB$
$LB, LA, \dots, \Box\neg LB$
$(\mid\sim^+)$ on $B \mid\sim A$
$(\mid\sim^+)$ on $B \mid\sim C$
$\dots, \neg LB, LB$
×
$\dots, \neg\Box\neg LB, \Box\neg LB$
×
$\dots, \neg LA, LA$
×
$\dots, \neg LB, LB$
×
$\dots, \neg\Box\neg LB, \Box\neg LB$
×
$\dots, \neg LC, LC$
×

Figure 5.9: A derivation of $\neg(A \mid\sim C), A \mid\sim B, B \mid\sim A, B \mid\sim C$ in $\mathcal{T}\mathbf{C}$. To save space, we use Δ to denote the set of positive conditionals $A \mid\sim B, B \mid\sim A, B \mid\sim C$.

$$(\mid\sim^+)\ \frac{\Gamma, A \mid\sim B}{\Gamma, A \mid\sim B, \neg LA \qquad \Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, A \mid\sim B, LA, \Box\neg LA \qquad \Gamma, A \mid\sim B, LA, \Box\neg LA, LB}$$

$$(\mid\sim^-)\ \frac{\Gamma, \neg(A \mid\sim B)}{LA, \Box\neg LA, \neg LB, \Gamma^{\mid\sim^{\pm}}} \qquad (L^-)\ \frac{\Gamma, \neg LA}{\Gamma^{L^{\downarrow}}, \neg A};\ \frac{\Gamma}{\Gamma^{L^{\downarrow}}}\ \text{if } \Gamma \text{ does not contain negated } L-\text{formulas}$$

Figure 5.10: Tableau calculus $\mathcal{T}\mathbf{C}$. Boolean rules are omitted.

Notice also that, as a difference with $\mathcal{T}\mathbf{CL}$, the $(\mid\sim^+)$ rule is neither a *static* nor a *dynamic* rule. Indeed, its leftmost and rightmost conclusions represent the same world as the world represented by the premise, whereas the conclusion in the middle represents a world which is different from the one represented by the premise. As an example, a derivation in $\mathcal{T}\mathbf{C}$ of the unsatisfiable set of formulas $\{\neg(A \mid\sim C), A \mid\sim B, B \mid\sim A, B \mid\sim C\}$ is presented in Figure 5.11.

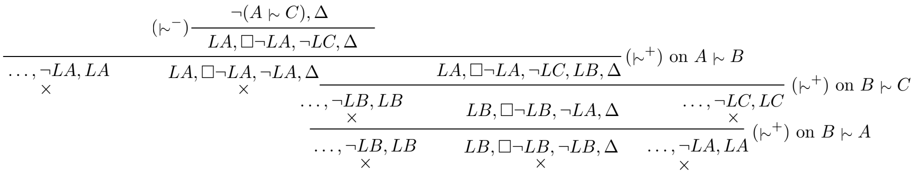

$\neg(A \mid\sim C), \Delta$
$(\mid\sim^-)$
$LA, \Box\neg LA, \neg LC, \Delta$
$(\mid\sim^+)$ on $A \mid\sim B$
$\dots, \neg LA, LA$
×
$LA, \Box\neg LA, \neg LA, \Delta$
×
$LA, \Box\neg LA, \neg LC, LB, \Delta$
$(\mid\sim^+)$ on $B \mid\sim C$
$\dots, \neg LB, LB$
×
$LB, \Box\neg LB, \neg LA, \Delta$
$\dots, \neg LC, LC$
×
$(\mid\sim^+)$ on $B \mid\sim A$
$\dots, \neg LB, LB$
×
$LB, \Box\neg LB, \neg LB, \Delta$
×
$\dots, \neg LA, LA$
×

Figure 5.11: A derivation in $\mathcal{T}\mathbf{C}$ of $\{\neg(A \mid\sim C), A \mid\sim B, B \mid\sim A, B \mid\sim C\}$. To save space, we use Δ to denote the set of positive conditionals $A \mid\sim B, B \mid\sim A, B \mid\sim C$.

We prove that $\mathcal{T}\mathbf{C}$ is sound and complete with respect to the semantics.

Theorem 5.33 (Soundness of $\mathcal{T}\mathbf{C}$)**.** *The system $\mathcal{T}\mathbf{C}$ is sound with respect to C-preferential models, i.e. given a set of formulas Γ, if there is a closed tableau for Γ, then Γ is unsatisfiable.*

Proof. Given a set of formulas Γ, if there is a closed tableau for Γ, then Γ is unsatisfiable in C-preferential models. For the rules already present in $\mathcal{T}\mathbf{CL}$, the proof is the same as the proof for Theorem 5.24 (notice that in the proof the transitivity of $<$ does not play any role). We only consider here the rule $(\mid\!\sim^+)$. We show that if the premise is satisfiable by a C-preferential model, then also one of the conclusions is.

Let $\mathcal{M}, w \models \Gamma, A \mid\!\sim B$. We distinguish the two following cases:

- $\mathcal{M}, w \not\models LA$, thus $\mathcal{M}, w \models \neg LA$: in this case, the left conclusion of the $(\mid\!\sim^+)$ rule is satisfied $(\mathcal{M}, w \models \Gamma, A \mid\!\sim B, \neg LA)$;
- $\mathcal{M}, w \models LA$: we consider two subcases:
  - $w \in Min_<(LA)$, hence $\mathcal{M}, w \models LA, \Box\neg LA$. By the definition of $\mathcal{M}, w \models A \mid\!\sim B$, we have that $\mathcal{M}, w \models LB$. Therefore, the right conclusion of $(\mid\!\sim^+)$ is satisfiable;
  - $w \not\in Min_<(LA)$: by the smoothness condition, there exists a world $w^{'} < w$ such that $w^{'} \in Min_<(LA)$. It follows that $(\mathcal{M}, w') \models LA, \Box\neg LA$. Furthermore, by the semantics of $\mid\!\sim$, $(\mathcal{M}, w') \models \Gamma^{\mid\sim^{\pm}}$ and since $w' < w$, $(\mathcal{M}, w') \models \Gamma^{\Box^{\downarrow}}$.

■

Soundness with respect to cumulative models follows from the correspondence established by Proposition 3.29.

We can prove that the (Weak-Cut) rule is admissible in $\mathcal{TC}$; this is stated by the following Theorem 5.40. In order to prove this Theorem, we need to prove some lemmas.

First of all, we prove that weakening is height-preserving admissible in our tableau calculi, i.e. if there is a closed tableau for a set of formulas Γ, then there is also a closed tableau for Γ, F (for any formula F of the language) of height no greater than the height of a tableau for Γ. Moreover, weakening is cut-preserving admissible, in the sense that the closed tableau for Γ, F does not add any application of (Weak-Cut) to the closed tableau for Γ. Furthermore, we prove that the rules for the boolean connectives are height-preserving and cut-preserving invertible, i.e. if there is a closed tableau for Γ, then there is a closed tableau for any set of formulas that can be obtained from Γ as a conclusion of an application of a boolean rule.

Lemma 5.34 (Height-preserving and cut-preserving admissibility of weakening). *Given a formula F, if there is a closed tableau of height h for Γ, then there is also a closed tableau for Γ, F of no greater height than h, i.e. weakening is height-preserving admissible. Moreover, the closed tableau for Γ, F does not add any application of (Weak-Cut) to the closed tableau for Γ, i.e. weakening is cut-preserving admissible.*

Proof. By induction on the height h of the closed tableau for Γ. The proof is easy and left to the reader.

Lemma 5.35 (Height-preserving and cut-preserving invertibility of boolean rules). *The rules for the boolean connectives are height-preserving invertible, i.e. given a set of formulas Γ and given any conclusion Γ', obtained by applying a boolean rule to Γ, if there is a closed tableau of height h for Γ, then there is a closed tableau for Γ' of height no greater than h. Moreover, the closed tableau for Γ' does not add any application of (Weak-Cut) to the closed tableau for Γ, i.e. the boolean rules are cut-preserving invertible.*

Proof. For each boolean rule (R), we proceed by induction on the height of the closed tableau for the premise. As an example, consider the $(\vee^+)$ rule. We show that, if $\Gamma, F \vee G$ has a closed tableau, also Γ, F and Γ, G have. If $\Gamma, F \vee G$ is an instance of the axiom (AX), then there is an atom P such that $P \in \Gamma$ and $\neg P \in \Gamma$, since axioms are restricted to atomic formulas only. Therefore, Γ has a closed tableau (it is an instance of (AX) too), and we conclude that both Γ, F and Γ, G have a closed tableau, since weakening is height-preserving admissible (Lemma 5.34 above). For the inductive step, we consider the first rule in the tableau for $\Gamma, F \vee G$. If $(\vee^+)$ is applied to $F \vee G$, then we are done, since we have closed tableaux for both Γ, F and Γ, G of a lower height than the premise's. If $(\mid\!\sim^-)$ is applied, then $F \vee G$ is removed from the conclusion, then Γ has a closed tableau; we conclude since weakening is height-preserving admissible (Lemma 5.34). If a boolean rule is applied, then $F \vee G$ is copied into the conclusion(s); in these cases, we can apply the inductive hypothesis and then conclude by re-applying the same rule. As an example, consider the case in which $(\rightarrow^-)$ is applied to $\Gamma', \neg(H \rightarrow I), F \vee G$ as follows:

$$\dfrac{\Gamma', \neg(H \rightarrow I), F \vee G}{(*)\Gamma', H, \neg I, F \vee G}\ (\rightarrow^-)$$

By the inductive hypothesis, there is a closed tableau (of no greater height than the height of $(*)$) for $(**)\Gamma', H, \neg I, F$ and for $(***)\Gamma', H, \neg I, G$. We conclude as follows:

$$\dfrac{\Gamma', \neg(H \rightarrow I), F}{(**)\Gamma', H, \neg I, F}\ (\rightarrow^-)$$

$$\dfrac{\Gamma', \neg(H \rightarrow I), G}{(***)\Gamma', H, \neg I, G}\ (\rightarrow^-)$$

If the first rule of the closed tableau for $\Gamma, F \vee G$ is $(\mid\!\sim^+)$ applied to $A \mid\!\sim B \in \Gamma$, then we have the following situation:

$$\dfrac{\Gamma, F \vee G}{(1)\Gamma, F \vee G, \neg LA \qquad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad (3)\Gamma, F \vee G, LA, \Box\neg LA, LB}\ (\mid\!\sim^+)$$

By the inductive hypothesis on (1), we have closed tableaux for $(1')\Gamma, F, \neg LA$ and for $(1'')\Gamma, G, \neg LA$. By the inductive hypothesis on (3), we have closed tableaux for $(3')\Gamma, F, LA, \Box\neg LA, LB$ and for $(3'')\Gamma, G, LA, \Box\neg LA, LB$. We conclude as follows:

$$\dfrac{\Gamma, F}{(1')\Gamma, F, \neg LA \qquad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad (3')\Gamma, F, LA, \Box\neg LA, LB}\ (\mid\!\sim^+)$$

$$\dfrac{\Gamma, G}{(1'')\Gamma, G, \neg LA \qquad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad (3'')\Gamma, G, LA, \Box\neg LA, LB}\ (\mid\!\sim^+)$$

If F (resp. G) were a conditional formula (even negated), we conclude as above, replacing the conclusion in the middle with $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, F, LA, \Box\neg LA$ (resp. $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, G, LA, \Box\neg LA$), for which there is a closed tableau since weakening is height-preserving admissible (Lemma 5.34).

■

Now we prove that we can assume, without loss of generality, that the conclusions of (Weak-Cut) are never derived by an application of $(\mid\!\sim^-)$, as stated by the following Lemma:

Lemma 5.36. *If* Γ *has a closed tableau, then there is a closed tableau for* Γ *in which all the conclusions of each application of (Weak-Cut) are derived by a rule different from* $(\mid\!\sim^-)$.

Proof. Consider an application of (Weak-Cut) in Γ in which one of its conclusions is obtained by an application of $(\mid\!\sim^-)$. The application of (Weak-Cut) is useless, since the premise of the cut can be obtained by applying directly $(\mid\!\sim^-)$ without (Weak-Cut). For instance, consider the following derivation, in which the conclusion in the middle of (Weak-Cut) is obtained by an application of $(\mid\!\sim^-)$:

$$\begin{array}{c} \Gamma', \neg(C \mid\!\sim D) \\ \hline \Gamma', \neg(C \mid\!\sim D), \neg LA \qquad \Gamma'^{\mid\sim\pm}, \Gamma'^{\Box\downarrow}, \neg(C \mid\!\sim D), LA, \Box\neg LA \qquad \Gamma', \neg(C \mid\!\sim D), \Box\neg LA \\ \hline (*)\Gamma'^{\mid\sim\pm}, LC, \Box\neg LC, \neg LD \end{array} \begin{array}{l} (\text{Weak-Cut}) \\ (\mid\!\sim^-) \end{array}$$

We can remove the application of (Weak-Cut), obtaining the following closed tableau:

$$\begin{array}{c} \Gamma', \neg(C \mid\!\sim D) \\ (*)\Gamma'^{\mid\sim\pm}, LC, \Box\neg LC, \neg LD \end{array} \; (\mid\!\sim^-)$$

Obviously, the proof can be concluded in the same way in the case the leftmost (resp. the rightmost) conclusion of (Weak-Cut) has a derivation starting with $(\mid\!\sim^-)$.

■

Now we prove that cut is admissible on propositional formulas and on formulas of the form LA. By cut we mean the following rule:

$$\begin{array}{c} \Gamma \\ \Gamma, F \qquad \Gamma, \neg F \end{array} \; (Cut)$$

and we show that it can be derived if F is a propositional formula or a formula of kind LA.

Lemma 5.37. *Given a set of propositional formulas* Γ *and a propositional formula* A*, if there is a closed tableau for both* $(1)\Gamma, \neg A$ *and* $(2)\Gamma, A$ *without (Weak-Cut), then there is also a closed tableau for* Γ *without (Weak-Cut).*

Proof. Since Γ is propositional, the only applicable rules are the propositional rules. The result follows by admissibility of cut in tableaux systems for propositional logic.

■

Lemma 5.38. *If there is a closed tableau without (Weak-Cut) for* $(1)\Gamma, \neg LA$ *and for* $(2)\Gamma, LA$*, then there is also a closed tableau without (Weak-Cut) for* Γ.

Proof. Let $h1$ be the height of the tableau for (1), and $h2$ the height of the tableau for (2). We proceed by induction on $h1 + h2$.
Base Case: $h1 + h2 = 0$. In this case, $h1 = 0$ and $h2 = 0$. In this case, (1) and (2) contain an axiom, and since axioms are restricted to atomic formulas, it can only be that $P, \neg P \in \Gamma$, where $P \in ATM$. Therefore, we conclude that there is a closed tableau for Γ without (Weak-Cut).

For the inductive step, we show that if the property holds in case $h1 + h2 = n - 1$, then it also holds in case $h1 + h2 = n$. We reason by cases according to which is the first rule applied to (1) or to (2). If the first rule applied to (1) is $(\mid\!\sim^-)$, applied to a conditional $\neg(C \mid\!\sim D) \in \Gamma$, let $\Gamma_{\neg(C \mid\sim D)}$ be the set of formulas so obtained. It can be easily verified that the same set can be obtained by applying the same rule to the same conditional in Γ, hence Γ has a closed tableau without (Weak-Cut). If the first rule applied to (1) is $(\mid\!\sim^+)$, applied to a conditional $(C \mid\!\sim D) \in \Gamma$, then we have that $(1a)\Gamma, \neg LC, \neg LA$, $(1b)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LC, \Box\neg LC$, and $(1c)$ $\Gamma, LC, \Box\neg LC, LD, \neg LA$ have a closed tableau with height smaller than $h1$. We can then apply weakening and the inductive hypothesis first over (2) and $(1a)$ and then over (2) and $(1c)$, to obtain that (i)$\Gamma, \neg LC$, and (ii) $\Gamma, LC, \Box\neg LC, LD$ respectively have a closed tableau without (Weak-Cut). Since (i),(1b),(ii) are obtained from Γ by applying $(\mid\!\sim^+)$ on $C \mid\!\sim D$, we conclude that also Γ has a closed tableau without (Weak-Cut). The case in which the first rule applied is a propositional rule immediately follows from the height-preserving invertibility of the boolean rules (see Lemma 5.35 above). For instance, suppose $(1)\Gamma', F \wedge G, \neg LA$ is derived by an application of $(\wedge^+)$, i.e. $(1')\Gamma', F, G, \neg LA$ has a closed tableau of height smaller than $h1$. Since $(2)\Gamma', F \wedge G, LA$ has a closed tableau, and $(\wedge^+)$ is height-preserving invertible, then also $(2')\Gamma', F, G, LA$ has a closed tableau of height no greater than $h2$, and we can conclude as follows:

$$\dfrac{\begin{array}{c}\Gamma', F \wedge G \\ \Gamma', F, G\end{array}\;(\wedge^+)}{(1')\Gamma', F, G, \neg LA \qquad (2')\Gamma', F, G, LA}\;(cut)$$

If the first rule applied to (2) is either a propositional rule, or $(\mid\!\sim^-)$, or $(\mid\!\sim^+)$, we can reason as for (1). We are left with the case in which the first rule applied both to (1) and to (2) is (L^-).

If the first rule applied to (1) is (L^-), applied to $\neg LB \in \Gamma$, let $\Gamma_{\neg LB}$ be the set obtained. The same set can be obtained by applying (L^-) to $\neg LB$ in Γ, hence Γ has a closed tableau without (Weak-Cut). If (L^-) is applied to $\neg LA$ itself, then $(*)\Gamma^{L^{\downarrow}}, \neg A$ has a closed tableau. In this case, we have to consider the tableau for (2). We distinguish two cases: in the first case (L^-) in (2) has been applied to some $\neg LB \in \Gamma$, in the second case Γ does not contain any negated L formula, hence (L^-) has not been applied to any specific $\neg LB$. First case: $(**)\Gamma^{L^{\downarrow}}, \neg B, A$ has a closed tableau. By weakening from $(*)$, also $(*')\Gamma^{L^{\downarrow}}, \neg B, \neg A$ has a closed tableau. By Lemma 5.37 applied to $(*')$ and $(**)$, also $\Gamma^{L^{\downarrow}}, \neg B$ has a closed tableau, and since this set can be obtained from Γ by applying (L^-) to $\neg LB$, it follows that Γ has a closed tableau, without (Weak-Cut). Second case: $(***)\Gamma^{L^{\downarrow}}, A$ has a closed tableau. By Lemma 5.37 applied to $(*)$ and $(***)$, also $\Gamma^{L^{\downarrow}}$ has a closed tableau, and since this set can be obtained by applying (L^-) to Γ, it follows that Γ has a closed tableau, without (Weak-Cut).

■

Lemma 5.39. *Let LA and LB be such that there is a closed tableau without (Weak-Cut) for $\{LA, \neg LB\}$. Then for all sets of formulas Γ, if $\Gamma, \Box\neg LA, \Box\neg LB$ has a closed tableau without (Weak-Cut), also $\Gamma, \Box\neg LB$ has.*

Proof. By induction on the height h of the tableau for $\Gamma, \Box\neg LA, \Box\neg LB$. If $h = 0$, then $\Gamma, \Box\neg LA, \Box\neg LB$ contains an axiom, hence (since axioms only concern atoms), also Γ does, and there is a tableau for $\Gamma, \Box\neg LB$ without (Weak-Cut). We prove that if the property holds for all tableaux of height $h-1$, then it also holds for tableaux of height h. We proceed by considering all possible cases corresponding to the first rule applied to $\Gamma, \Box\neg LA, \Box\neg LB$.

The case in which the first rule is boolean is easy and left to the reader.

If the first rule is (L^-), it can be easily verified that the same set of formulas can be obtained by applying (L^-) to $\Gamma, \Box\neg LB$ that hence has a closed tableau without (Weak-Cut).

If the first rule is $(\mid\!\sim^-)$, again it can be easily verified that the same set of formulas can be obtained by applying $(\mid\!\sim^-)$ to $\Gamma, \Box\neg LB$, that hence has a closed tableau without (Weak-Cut).

If the first rule is $(\mid\!\sim^+)$ applied to a conditional $C \mid\!\sim D \in \Gamma$, then $(1)\Gamma, \Box\neg LA, \Box\neg LB, \neg LC$; $(2)\ \Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \neg LA, \neg LB, \Box\neg LC, LC$; $(3)\ \Gamma, \Box\neg LA, \Box\neg LB, \Box\neg LC, LC, LD$ have a closed tableau with height smaller than h. By the inductive hypothesis from (1) and (3), we infer that $(1')\Gamma, \Box\neg LB, \neg LC$ and $(3')\Gamma, \Box\neg LB, \Box\neg LC, LC, LD$ have a closed tableau without (Weak-Cut). Furthermore, since by hypothesis $\{LA, \neg LB\}$ has a closed tableau without (Weak-Cut), from (2), weakening (Lemma 5.34), and Lemma 5.38, we infer that also $(2')\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LC, \Box\neg LC, \neg LB$ has a closed tableau without (Weak-Cut). Since $(1'), (2'), (3')$ can be obtained from $\Gamma, \Box\neg LB$ by applying $\mid\!\sim^+$ to $C \mid\!\sim D$ in Γ, we conclude that $\Gamma, \Box\neg LB$ has a closed tableau without (Weak-Cut).

There are no other cases, hence the result follows.

■

Now we are able to prove that the (Weak-Cut) is admissible in $\mathcal{T}\mathbf{C}$, as stated by Theorem 5.40:

Theorem 5.40. *Given a set of formulas Γ and a propositional formula A, if there is a closed tableau for each of the following sets of formulas:*

(1) $\Gamma, \neg LA$

(2) $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA$

(3) $\Gamma, \Box\neg LA$

then there is also a closed tableau for Γ, i.e. the (Weak-Cut) *rule is admissible.*

Proof. We prove that for all sets of formulas Γ, if there is a closed tableau without (Weak-Cut) for each of the following sets of formulas:

(1) $\Gamma, \neg LA$

(2) $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA$

(3) $\Gamma, \Box\neg LA$

then there is also a closed tableau without (Weak-Cut) for Γ. By this property, given a closed tableau for a starting set of formulas Γ_0, a closed tableau without (Weak-Cut) for Γ_0 can be obtained by eliminating all applications of (Weak-Cut), starting from the leafs towards the root of the tableau.

First of all, notice that in general there can be several closed tableaux for Γ_0. We only consider closed tableaux for Γ_0 that are *minimal* (in the number of nodes). Moreover,

by Lemma 5.36 we can restrict our concern to applications of (Weak-Cut) whose conclusions are not obtained by applications of $(\mid\!\sim^-)$.
Let $h1, h2, h3$ be the heights of the tableaux for (1), (2), and (3) respectively. We proceed by induction on $h1 + h2 + h3$. The reader might be surprised by the fact that the proof is carried on by single induction on the heights of the premises of (Weak-Cut). Indeed, the proof relies on the proof of admissibility of ordinary cut at the propositional level (Lemma 5.37), that is proved as usual by double induction on the complexity of the cut formula and on the heights of the derivation of the two conclusions.

For the base case, notice that always $h2 > 0$, since axioms are restricted to atomic formulas, and $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$ by definition does not contain any atomic formula. Our base case will hence be the case in which $h1 = 0$ or $h3 = 0$, i.e. (1) or (3) are axioms, and $h2$ is minimal.
Base Case: $h1 = 0$, $h3 = 0$, and $h2$ is minimal. If $h1 = 0$, then (1) is an axiom. In this case, since axioms restricted to atomic formulas, it must be that $P, \neg P \in \Gamma$ with $P \in ATM$. Therefore we can conclude that there is a closed tableau for Γ without (Weak-Cut).
For the inductive step, we distinguish the two following cases:

1. one of the conclusions of (Weak-Cut) is obtained by an application of a rule for the boolean connectives;

2. all the conclusions of (Weak-Cut) are obtained by $(\mid\!\sim^+)$ or by (L^-).

The list is exhaustive; indeed, by Lemma 5.36, we can consider, without loss of generality, a closed tableau in which all the conclusions of each application of (Weak-Cut) are obtained by a rule different from $(\mid\!\sim^-)$.

We consider the two cases above:

1. rules for the boolean connectives: first, notice that a boolean rule cannot be applied to the conclusion in the middle of (Weak-Cut), since it only contains conditional formulas (even negated), L- formulas (even negated), and a positive box formula. In these cases, we conclude by permuting the (Weak-Cut) rule over the boolean rule, i.e. we first cut the conclusion(s) of the boolean rule with the other conclusions of (Weak-Cut), then we conclude by applying the boolean rule on the sets of formulas obtained. As an example, consider the following closed tableau, where the rightmost conclusion of (Weak-Cut) is obtained by an application of $(\vee^+)$, and where F and G are conditional formulas:

$$\dfrac{\dfrac{\Gamma', F \vee G}{(1)\Gamma', F \vee G, \neg LA \qquad (2)\Gamma'^{\mid\sim^{\pm}}, \Gamma'^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad (3)\Gamma', F \vee G, \Box\neg LA}\ (\text{Weak-Cut})}{(3a)\Gamma', F, \Box\neg LA \qquad (3b)\Gamma', G, \Box\neg LA}\ (\vee^+)$$

Since $(\vee^+)$ is height-preserving and cut-preserving invertible (see Theorem 5.35), there is a closed tableau of no greater height than (1), having no applications of (Weak-Cut) (since the closed tableau starting with (1) does not contain any application of it), of $(1')\Gamma', F, \neg LA$ and $(1'')\Gamma', G, \neg LA$. By the height-preserving admissibility of weakening (see Lemma 5.34), we have also a closed tableau, of no greater height than (2), for $(2')\Gamma'^{\mid\sim^{\pm}}, \Gamma'^{\Box^{\downarrow}}, F, LA, \Box\neg LA$ and for

$(2'')\Gamma'^{\mid\sim^{\pm}}, \Gamma'^{\Box^{\downarrow}}, G,\ LA, \Box\neg LA$. We can apply the inductive hypothesis to $(1')$, $(2')$, and $(3a)$ to obtain a closed tableau, without applications of (Weak-Cut), of $(*)\Gamma', F$. Notice that we can apply the inductive hypothesis since the tableaux for $(1')$ and $(2')$ do not contain applications of (Weak-Cut) and have no greater height than (1) and (2), respectively; however, the closed tableau for $(3a)$, not containing any application of (Weak-Cut), has obviously a smaller height than the one for (3). In the same way, we can apply the inductive hypothesis to $(1'')$, $(2'')$, and $(3b)$ to obtain a closed tableau for $(**)\Gamma', G$. We can conclude by applying $(\vee^+)$ to $(*)$ and $(**)$ to obtain a proof without (Weak-Cut) for $\Gamma', F \vee G$.

2. $(\mid\sim^+)$ or (L^-): let us denote with a triple $< R_1, R_2, R_3 >$ the rules applied, respectively, to the three conclusions (1), (2), and (3) of (Weak-Cut), where $R_i \in \{(L^-), (\mid\sim^+), (\text{ANY})\}$. We use (ANY) to represent either (L^-) and $(\mid\sim^+)$. As an example, $< (L^-), (\mid\sim^+), (\text{ANY}) >$ is used to represent the case in which the leftmost conclusion of (Weak-Cut) (1) is obtained by an application of (L^-), whereas the conclusion in the middle (2) is obtained by an application of $(\mid\sim^+)$; the rightmost conclusion (3) can be either obtained by an application of (L^-) or by an application of $(\mid\sim^+)$.

   We distinguish the following cases: (†) $< (\text{ANY})(\text{ANY})(L^-) >$, (††) $< (\text{ANY}), (L^-), (\mid\sim^+) >$, and (†††) $< (\text{ANY}), (\mid\sim^+), (\mid\sim^+) >$. The list is exhaustive.

   - (†) $< (\text{ANY})(\text{ANY})(L^-) >$: the tableau for (3) is started with an application of (L^-) to a formula $\neg LB \in \Gamma$ $(\Gamma = \Gamma', \neg LB)$:

     $$\cfrac{(3)\Gamma', \neg LB, \Box\neg LA}{\Gamma'^{L^{\downarrow}}, \neg B}\ (L^-)$$

     We can conclude since $\Gamma'^{L^{\downarrow}}, \neg B$ can be obtained by applying (L^-) to $\Gamma = \Gamma'.\neg LB$ (the formula $\Box\neg LA$ is removed from the conclusion in an application of (L^-)). In case the tableau for (3) is started with an application of (L^-) on a formula $LB \in \Gamma$, and there is no $\neg LB_i \in \Gamma$, we can obviously conclude in the same manner;

   - (††) $< (\text{ANY}), (L^-), (\mid\sim^+) >$: the proof of (3) is started with an application of $(\mid\sim^+)$ on a formula $C \mid\sim D \in \Gamma$; we have that the following sets of formulas have three closed tableaux without (Weak-Cut):

     - (3a) $\Gamma, \Box\neg LA, \neg LC$
     - (3b) $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \neg LA, \Box\neg LC, LC$
     - (3c) $\Gamma, \Box\neg LA, \Box\neg LC, LC, LD$

     Moreover, since the first rule applied to (2) is (L^-), we have that $\{LA, \Gamma^{\Box^{\downarrow}}\}$ has a closed tableau without (Weak-Cut).

     We want to show that also the three following sets of formulas:

     - (I) $\Gamma, \neg LC$
     - (II) $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \Box\neg LC, LC$
     - (III) $\Gamma, \Box\neg LC, LC, LD$

     have a closed tableau without (Weak-Cut), from which we conclude by an application of $(\mid\sim^+)$. We thus want to show that:

(I). By (3a) we know that $\Gamma, \neg LC, \Box\neg LA$ has a closed tableau without (Weak-Cut); by weakening from (1) we also know that $\Gamma, \neg LC, \neg LA$ has a closed tableau without (Weak-Cut); last, by (2) we know that $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $\Box\neg LA$, LA has a closed tableau without (Weak-Cut). Since the sum of the heights of (1), (2) and (3a) is smaller than h1 + h2 + h3 (because the height of (3a) is smaller than h3), we can apply the inductive hypothesis and conclude that (I) $\Gamma, \neg LC$ has a closed tableau without (Weak-Cut).

(II) Since $\{LA, \Gamma^{\Box^{\downarrow}}\}$ has a closed tableau without (Weak-Cut), by weakening, also $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LC, LC$ has a closed tableau without (Weak-Cut), and from (3b) and Lemma 5.38, also (II) $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \Box\neg LC, LC$ has.

(III). By weakening from (1), we know that $\Gamma, LC, \Box\neg LC, LD$, $\neg LA$ has a closed tableau without (Weak-Cut); by (3*c*) we know that Γ, LC, $\Box\neg LC$, $LD, \Box\neg LA$ has a closed tableau without (Weak-Cut), and by weakening from (2) we know that $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $\neg LC$, $\Box\neg LA, LA$ has a closed tableau without (Weak-Cut). Furthermore, the sum of the heights of the tableaux for (1), (2) and (3c) is smaller than h1+h2+h3. We can then apply the inductive hypothesis to conclude that (III): $\Gamma, \Box\neg LC, LC, LD$ has a closed tableau without (Weak-Cut).

- (†††) $<$ (ANY), $(\mid\!\sim^{+}), (\mid\!\sim^{+})$ $>$: as in the previous case, the proof of (3) is started with an application of $(\mid\!\sim^{+})$ on a formula $C \mid\!\sim D \in \Gamma$; we have that the following sets of formulas have three closed tableaux without (Weak-Cut):

  - (3a) $\Gamma, \Box\neg LA, \neg LC$
  - (3b) $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \neg LA, \Box\neg LC, LC$
  - (3c) $\Gamma, \Box\neg LA, \Box\neg LC, LC, LD$

  Moreover, we have that the first rule applied in (2) is $(\mid\!\sim^{+})$ applied to some conditional $C_1 \mid\!\sim D_1 \in \Gamma^{\mid\!\sim^{\pm}}$. We show that there is a conditional $C_i \mid\!\sim D_i \in \Gamma$ such that:

  - (*i*) $\Gamma, \neg LC_i$
  - (*ii*) $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \Box\neg LC_i, LC_i$
  - (*iii*) $\Gamma, \Box\neg LC_i, LC_i, LD_i$

  have a closed tableau without (Weak-Cut), hence also Γ has, since (*i*), (*ii*), and (*iii*) can be obtained from Γ by applying $(\mid\!\sim^{+})$ to $C_i \mid\!\sim D_i$.

  The tableau for (2) can contain a sequence s of applications of $(\mid\!\sim^{+})$. By considering only the leftmost branch introduced by the applications of $(\mid\!\sim^{+})$ in s, consider the last application of $(\mid\!\sim^{+})$ to a conditional $C_i \mid\!\sim D_i$. We will have that (2a) $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA$, $\Box\neg LA, \neg LC_1, \ldots, \neg LC_i$ has a closed tableau; (2b) $\Gamma^{\mid\!\sim^{\pm}}, \neg LA$, $\Box\neg LC_i, LC_i$ has a closed tableau; (2c) $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $LA, \Box\neg LA, \neg LC_1$, $\ldots, \neg LC_{i-1}, \Box\neg LC_i, LC_i, LD_i$ has a closed tableau. The situation can be represented as follows:

$$\begin{array}{c}
\dfrac{\dfrac{\Gamma}{(1)\Gamma, \neg LA \qquad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \qquad (3)\Gamma, \Box\neg LA}\,(\text{Weak-Cut})}{\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1 \quad \dots}\,(\mid\sim^{+}) \\
\vdots \\
(2a)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \dots, \neg LC_i \quad \dots (2b) \dots (2c) \dots
\end{array}$$

Furthermore, since by hypothesis the tableau for (2) does not contain any application of (Weak-Cut), also the tableaux for (2a), (2b), and (2c) do not contain any application of (Weak-Cut).

Observe that (2a) $\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \dots, \neg LC_i$ cannot be an instance of (AX), since axioms are restricted to atomic formulas only, and (2a) only contains conditionals, L-formulas, and boxed formulas. Therefore, we can observe that the rule applied to (2a) is (L^-): indeed, if the rule was $(\mid\sim^-)$ applied to some $\neg(C_k \mid\sim C_j) \in \Gamma^{\mid\sim^{\pm}}$, then we could find a shorter derivation, obtained by applying $(\mid\sim^-)$ to $(2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA$, against the minimality of the closed tableau we are considering. This situation would be as follows:

$$\begin{array}{c}
\dfrac{\dfrac{\Gamma}{(1)\Gamma, \neg LA \quad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \quad (3)\Gamma, \Box\neg LA}\,(\text{Weak-Cut})}{\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1 \quad \dots}\,(\mid\sim^{+}) \\
\vdots \\
\dfrac{\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \dots, \neg LC_i \quad \dots}{(\Gamma^{\mid\sim^{\pm}} - \{\neg(C_k \mid\sim C_j)\}), LC_k, \Box\neg LC_k, \neg LC_j}\,(\mid\sim^{-})
\end{array}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\dfrac{\dfrac{\Gamma}{(1)\Gamma, \neg LA \qquad (2)\Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA \quad (3)\Gamma, \Box\neg LA}\,(\text{Weak-Cut})}{\dots \qquad (\Gamma^{\mid\sim^{\pm}} - \{\neg(C_k \mid\sim C_j)\}), LC_k, \Box\neg LC_k, \neg LC_j \quad \dots}\,(\mid\sim^{-})$$

More precisely, we can observe that the rule applied to (2a) is (L^-) applied to $\neg LC_i$: indeed, if (L^-) was applied to a previously generated negated formula, there would be a shorter tableau obtained by immediately applying (L^-) to that formula, as represented by the following derivations:

$$\vdots$$

$$\dfrac{\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \ldots, \neg LC_h}{\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \ldots, \neg LC_h, \neg LC_{h+1} \quad \ldots} \ (\mid\!\sim^{+})$$

$$\vdots$$

$$\dfrac{\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \ldots, \neg LC_h, \ldots, \neg LC_i \qquad \ldots}{A, \neg C_h} \ (L^{-})$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\vdots$$

$$\begin{array}{c} \Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, LA, \Box\neg LA, \neg LC_1, \ldots, \neg LC_h \\ \qquad\qquad\qquad\qquad\qquad\qquad (L^{-}) \\ A, \neg C_h \end{array}$$

Hence, $A, \neg C_i$ has a closed tableau without (Weak-Cut), and hence also $(*) LA, \neg LC_i$ has a closed tableau without (Weak-Cut). From $(*)$ and (1), with opportune weakenings, by Lemma 5.38 we derive that $(i) : \Gamma, \neg LC_i$ has a closed tableau without (Weak-Cut). By weakening from $(2b)$, we have that $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $\neg LA, \Box\neg LC_i, LC_i$ has a closed tableau without (Weak-Cut). From this set of formulas, (2), weakening, and Lemma 5.38, we also know that: $\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $\Box\neg LA, \Box\neg LC_i, LC_i$ has a closed tableau without (Weak-Cut). From this set of formulas, (*) and Lemma 5.39, we conclude that also $(ii)\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, \Box\neg LC_i, LC_i$ has a closed tableau without (Weak-Cut).

Consider now (iii): we can show that $\Gamma, \Box\neg LC_i, LC_i, LD_i$ has a closed tableau without (Weak-Cut). Indeed, we can repeat the same proofs of case (††) in order to show that (I) $\Gamma\neg LC$ and (III) $\Gamma, \Box\neg LC, LC, LD$ have a closed tableau without (Weak-Cut) (in (††) there was no assumption on the first rule applied to (2) to show that (I) and (III) have a closed tableau). In order to show that (iii) has a closed tableau without (Weak-Cut), we observe that: by weakening from (I), (I')$\Gamma, \Box\neg LC_i, LC_i, LD_i, \neg LC$ has a closed tableau without (Weak-Cut); by weakening from (III), (III')$\Gamma, \Box\neg LC_i, LC_i, LD_i, \Box\neg LC$, LC, LD has a closed tableau without (Weak-Cut); since (3b) and (*) have closed tableaux without (Weak-Cut), by Lemma 5.38 we have that (II')$\Gamma^{\mid\!\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}$, $\neg LC_i, \Box\neg LC, LC$ has a closed tableau without (Weak-Cut). We conclude that $(iii)\Gamma$, $LC_i, \Box\neg LC_i$, LD_i has a closed tableau without (Weak-Cut), obtained by applying $(\mid\!\sim^{+})$ to (I'), (II'), and (III').

We have hence proven that $(i), (ii)$, and (iii) have a closed tableau without (Weak-Cut), then we can conclude by applying $(\mid\!\sim^{+})$ to them to obtain a closed tableau for Γ.

■

The completeness of the calculus can be proven by modifying the model construction used to show the completeness of $\mathcal{T}\mathbf{CL^T}$ (Theorem 5.25). The completeness of $\mathcal{T}\mathbf{C}$ is a consequence of the admissibility of the (Weak-Cut) rule. Hence, in the construction of the model, we make use of the (Weak-Cut) rule. In order to build a *finite* model for an initial set of formulas Γ, we essentially restrict the application of (Weak-Cut) to formulas built from the finite language of Γ and we control the application of the rule in order to avoid to apply this rule to propositionally equivalent formulas.

Theorem 5.41 (Completeness of $\mathcal{T}$**C**). *$\mathcal{T}$**C** is complete with respect to C-preferential models, i.e. if a set of formulas Γ is unsatisfiable, then Γ has a closed tableau in $\mathcal{T}$**C**.*

Proof. We assume that no tableau for Γ is closed, and we construct a model for Γ. The completeness is a consequence of the admissibility of (Weak-Cut). Hence, in the construction of the model, we make use of (Weak-Cut), i.e. we consider $\mathcal{T}$**C**+ (Weak-Cut). This is used to ensure the smoothness condition in the resulting model. In order to keep the construction of the model finite, we define a notion of equivalence between formulas with respect to their propositional part (or *p-equivalence*) so to identify those formulas having the same structure and containing equivalent propositional components. Let $\equiv_{PC}$ be logical equivalence in the classical propositional calculus. We define the notion of *p-equivalence* between two formulas F and G (written $F \equiv_p G$) as an equivalence relation satisfying the following conditions:

- if F and G are propositional formulas, then $F \equiv_p G$ iff $F \equiv_{PC} G$;
- $LA \equiv_p LB$ iff $A \equiv_{PC} B$;
- $\neg LA \equiv_p \neg LB$ iff $A \equiv_{PC} B$;
- $\Box\neg LA \equiv_p \Box\neg LB$ iff $A \equiv_{PC} B$.

For instance, $LA \equiv_p L(A \wedge A)$, $\Box\neg LA \equiv_p \Box\neg L(A \wedge A)$.

We say that two sets of formulas Γ_i and Γ_j are p-equivalent if for every formula in Γ_i there is a p-equivalent formula in Γ_j, and *vice versa*.

Observe that this notion of p-equivalence is very weak and, for instance, we do not recognize that the set $\{LA, LB\}$ is equivalent to the set $\{L(A \wedge B)\}$. The notion of p-equivalence has been introduced with the purpose of limiting the application of the rule (Weak-Cut) so that it will not introduce infinitely many equivalent formulas. Moreover, in the construction of the model, we will not add a set of formulas to the current set of worlds X if X already contains p-equivalent set of formulas.

In our construction of the model below we will identify p-equivalent sets of formulas Γ_i. Before adding a new set of formulas Γ_i to the current set of worlds X, we check that X does not already contain a node Γ_j such that Γ_j is p-equivalent to Γ_i; for short, we will write $\Gamma_i \notin_P X$ in case X does not already contain such a Γ_j.

We define the procedure `SAT'` that extends any Γ_i by applying the transformations described below and, at the same time, produces a set Γ_i^S, initially set to $\emptyset$. Γ_i^S is a set of non saturated worlds. These worlds will then be processed at their turn in the model construction. The reason why `SAT'` is different from `SAT` previously defined is that, differently from `SAT`, it has to cope with rules (such as ($\mid\!\sim^+$) or (Weak-Cut)) which are neither static nor dynamic. The transformations below are performed in sequence:

- apply to Γ_i the propositional rules, once to each formula, as far as possible. In case of branching, make the choice that leads to an open tableau (this step saturates Γ_i with respect to the static rules);

- for each $A \mid\!\sim B \in \Gamma_i$, apply $(\mid\!\sim^+)$ to it. If the leftmost branch is open, then add $\neg LA$ to Γ_i; otherwise, if the rightmost branch is open, then add $\Box\neg LA, LA, LB$ to Γ_i; if the only open branch is the one in the middle, then add the set $\Delta = \{\Gamma_i^{\mid\sim^\pm}, \Gamma_i^{\Box^\downarrow}, LA, \Box\neg LA\}$ to the support set Γ_i^S associated with Γ_i;
- for all $LA \in \mathcal{L}_\Gamma$, if there is no $A \mid\!\sim B \in \Gamma_i$, and there is no A' propositionally equivalent to A on which (Weak-Cut) has already been applied (in Γ_i), apply (Weak-Cut) to it. If the leftmost branch is open, then add $\neg LA$ to Γ_i; otherwise, if the rightmost branch is open, add $\Box\neg LA$ to Γ_i; if the only open branch is the one in the middle, then add $\Delta = \{\Gamma_i^{\mid\sim^\pm}, \Gamma_i^{\Box^\downarrow}, LA, \Box\neg LA\}$ to the support set Γ_i^S.

Observe that `SAT'` terminates, extends Γ_i to a pair of finite sets of formulas, and produces a set Γ_i^S which is a finite set of finite sets of formulas, since: (a) there is only a finite number of conditionals that can lead to create sets of formulas in Γ_i^S; (b) there is only a finite number of formulas which are not $p-$equivalent with each other and that can lead to create a set in Γ_i^S by the third transformation above.
We build X, the set of worlds of the model, and $<$, as follows:

```
1. initialize $X = \{\Gamma\}$; mark $\Gamma$ as unresolved;
2. **while** $X$ contains unresolved nodes **do**
   3. choose an unresolved $\Gamma_i$ from $X$;
   4. **for** each $\Delta \in \Gamma_i^S$, associated with $\Gamma_i$,
          let $\Gamma_\Delta = \texttt{SAT'}(\Delta)$;
      4a. **for** all $\Gamma_j \in X$ s.t. $\Gamma_j$ is p-equivalent to $\Gamma_\Delta$
          add the relation $\Gamma_j < \Gamma_i$;
      4b. **if** $\Gamma_\Delta \notin_P X$ **then** let $X = X \cup \{\Gamma_\Delta\}$ and mark $\Gamma_\Delta$ as unresolved;
   5. **for** each formula $\neg LA \in \Gamma_i$, let $\Gamma_{i,\neg LA} = \texttt{SAT'}(\texttt{APPLY}(\Gamma_i, L^-, \neg LA))$;
      5a. **for** all $\Gamma_j \in X$ s.t. $\Gamma_j$ is p-equivalent to $\Gamma_{i,\neg LA}$
          add the relation $(\Gamma_j, \Gamma_i)$ to $R$;
      5b. **if** $\Gamma_{i,\neg LA} \notin_P X$ **then** let $X = X \cup \{\Gamma_{i,\neg LA}\}$ and mark $\Gamma_{i,\neg LA}$ as unresolved;
   6. **for** each formula $\neg(A \mid\!\sim B) \in \Gamma_i$
      6a. let $\Gamma_{i,\neg(A\mid\sim B)} = \texttt{SAT'}(\texttt{APPLY}(\Gamma_i, \mid\!\sim^-, \neg(A \mid\!\sim B)))$;
      6b. **if** $\Gamma_{i,\neg(A\mid\sim B)} \notin_P X$ **then** $X = X \cup \{\Gamma_{i,\neg(A\mid\sim B)}\}$ and mark $\Gamma_{i,\neg(A\mid\sim B)}$ as unresolved;
   7. mark $\Gamma_i$ as resolved;
**endWhile**;
```

If Γ is finite, the procedure terminates. Indeed, it can be seen that `SAT'` terminates, as there is only a finite number of propositional evaluations. Furthermore, the whole procedure terminates, since the number of possible different sets of formulas that can be added to X starting from a finite set Γ is finite. Indeed, the number of non-p-equivalent sets of formulas that can be introduced in X is finite, as the number of p-equivalent classes is finite.

We construct the model $\mathcal{M} = \langle X, R_X, <_X, V\rangle$ as follows.

- X,V and R_X are defined as in the completeness proof for $\mathcal{T}$**CL** (Theorem 5.25);
- $<_X$ is defined as follows:
  - (i) if $\Gamma' < \Gamma$, then $\Gamma' <_X \Gamma$;
  - (ii) if $\Gamma' < \Gamma$, and $\Gamma R_X \Gamma''$, then $\Gamma' <_X \Gamma''$;

  As a difference from $<_X$ used in the completeness proof for $\mathcal{T}$**CL** (Theorem 5.25), $<_X$ is not transitive.

In order to show that $\mathcal{M}$ is a **C**-preferential model for Γ, we prove the following facts:

Fact 5.42. *The relation $<_X$ is irreflexive.*

Proof of Fact 5.42. By the procedure above, $\Gamma_j <_X \Gamma_i$ only in two cases: 1) the relation has been introduced by step 4a in the procedure. In this case, it can be seen that $\Gamma_i \neq \Gamma_j$. 2) the relation has been introduced when completing $<_X$ starting from $<$, i.e. $\Gamma_j < \Gamma_k$ and $\Gamma_k R_X \Gamma_i$. Also in this case, it can be seen that $\Gamma_i \neq \Gamma_j$, since Γ_i does not contain L-formulas, whereas Γ_j does.

□ *Fact 5.42*

By reasoning analogously to what done for Facts 5.10 and 5.27, we show that:

Fact 5.43. *For all formulas F and for all $\Gamma_i \in X$ we have that:*
(i) if $F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \models F$; (ii) if $\neg F \in \Gamma_i$ then $\mathcal{M}, \Gamma_i \not\models F$.

Proof of Fact 5.43. The proof is very similar to the one of Facts 5.10 and 5.27 for **CL**. Obviously, the case of negated boxed formulas disappears. Here we only consider the case of positive conditional formulas since the rule $(\mid\!\sim^+)$ in $\mathcal{TC}$ is slightly different than before. Let $\Delta \in Min_{<_X}(LA)$. We distinguish two cases:

- $A \mid\!\sim B \in \Delta$. By definition of SAT' and construction of the model above, either (1)$\neg LA \in \Delta$ or (2) there is $\Delta_{LA} \in X$ such that $LA \in \Delta_{LA}$ and $\Delta_{LA} < \Delta$, hence $\Delta_{LA} <_X \Delta$ or (3)$LB \in \Delta$. Similarly to what done in the proof for Fact 5.27, (1) and (2) cannot be the case, since they both contradict the fact that $\Delta \in Min_{<_X}(LA)$. Thus it must be that (3)$LB \in \Delta$, and by inductive hypothesis $\mathcal{M}, \Delta \models LB$.
- $A \mid\!\sim B \not\in \Delta$. We can reason in the same way than in the analogous case in the proof of Fact 5.27 above: Δ must have been generated by applying (L^-) to Δ' with $A \mid\!\sim B \in \Delta'$, hence $\Delta' R_X \Delta$. By definition of SAT' above, either (1)$\neg LA \in \Delta'$ or (2) there is $\Delta'_{LA} \in X$ such that $LA \in \Delta'_{LA}$ and $\Delta'_{LA} < \Delta'$, or (3) $LB \in \Delta'$. (1) is not possible: by inductive hypothesis, it would be $\mathcal{M}, \Delta' \not\models LA$, i.e. there is $\Delta^{''}$ such that $\Delta R_X \Delta^{''}$ and $\mathcal{M}, \Delta^{''} \not\models A$. By definition of R_X (see point (ii) in the definition of R_X, proof of Theorem 5.25), also $\Delta R_X \Delta^{''}$, hence also $\mathcal{M}, \Delta \not\models LA$, which contradicts $\Delta \in Min_{<_X}(LA)$. If (2), by definition of $<_X$, $\Delta'_{LA} <_X \Delta$, which contradicts $\Delta \in Min_{<_X}(LA)$. It follows that $LB \in \Delta'$, hence by inductive hypothesis $\mathcal{M}, \Delta' \models LB$, and also $\mathcal{M}, \Delta \models LB$ (indeed, since Δ does not contain any $L-$ formula, by construction of the model and by definition of R_X - see point (ii) in the definition of R_X, proof of Theorem 5.25 - $\Delta R_X \Delta^{''}$ just in case $\Delta' R_X \Delta^{''}$, from which the result follows).

□ *Fact 5.43*

Furthermore, we prove that:

Fact 5.44. *The relation $<$ satisfies the smoothness condition on L-formulas.*

Proof of Fact 5.44. Let $\mathcal{M}, \Gamma_i \models LA$. Then by Fact 5.43 above, $\neg LA \notin \Gamma_i$. By definition of `SAT'` and point 4 in the procedure above, either $\Box\neg LA \in \Gamma_i$ or there is $\Gamma_j \in X$ s.t. $\{LA, \Box\neg LA\} \subseteq \Gamma_j$, and $\Gamma_j <_X \Gamma_i$. In the first case, by Fact 2 above, $\mathcal{M}, \Gamma_i \models \Box\neg LA$, and it is minimal with respect to the set of LA-worlds. In the second case $\mathcal{M}, \Gamma_j \models \Box\neg LA$, it is minimal with respect to the set of LA-worlds, and $\Gamma_j <_X \Gamma_i$.

□ *Fact 5.44*

From the above facts, we can conclude that $\mathcal{M}$ is a **C**-preferential model for Γ, which concludes the completeness proof.

■

From the above facts, we can conclude that $\mathcal{M}$ is a **C**-preferential model for Γ, which concludes the completeness proof. From the above Theorem 5.41, together with Proposition 3.29, it follows that for any boolean combination of conditionals Γ, if it does not have any closed tableau, then it is satisfiable in a cumulative model.

Similarly to what done for **P** and **CL**, we can show the following Corollary.

Corollary 5.45 (Finite model property). **C** *has the finite model property.*

By Theorem 5.40, only formulas occurring in the initial set Γ can occur on a branch. Hence, the number of possibly different sets of formulas Γ on the branch is finite (and they are exponentially many in the size of Γ). A loop checking procedure can be used in order to avoid that a given set of formulas is expanded again on a branch, so to ensure the termination of the procedure.

To check wether Γ is satisfiable, we must check that all the tableaux for Γ have an open branch. As there are exponentially many tableaux that have to be taken into consideration, each one of exponential size with respect to the size of the initial set of formulas, our tableau method provides immediately an hyper exponential procedure to check the satisfiability. In further investigations it might be considered if this bound can be improved. For this, a more accurate analysis of the structure of a derivation (and, in particular, an analysis of permutability of the rules) might be required.

5.5 A Tableau Calculus for Rational Logic R

In this section we present $\mathcal{T}$**R**, a tableau calculus for rational logic **R**. We have already mentioned that, as a difference with the calculi presented for the other weaker logics, the calculus for **R** is a *labelled* calculus; the use of labels seems ta more natural approach. Indeed, in order to capture the modularity condition of the preference relation, intuitively we must keep all worlds generated by $(\Box^-)$ and we need to propagate formulas among them according to all possible modular orderings. In an unlabelled calculus, this might be achieved, for instance, by introducing an ad hoc modal operator (that acts as a marker) or by adding additional structures to tableau nodes similarly to hypersequents calculi (see for instance [Avr96]). However, the resulting calculus would be unavoidably rather cumbersome. In contrast, by using world labels, we can easily keep track of multiple worlds and their relations. This provides a much

$$(\mathbf{AX})\ \Gamma, x : P, x : \neg P \quad \text{with } P \in ATM \qquad\qquad (\mathbf{AX})\ \Gamma, x < y, y < x$$

$$(\neg)\ \frac{\Gamma, x : \neg\neg F}{\Gamma, x : F} \qquad (\wedge^+)\ \frac{\Gamma, x : F \wedge G}{\Gamma, x : F, x : G} \qquad (\wedge^-)\ \frac{\Gamma, x : \neg(F \wedge G)}{\Gamma, x : \neg F \qquad\qquad \Gamma, x : \neg G}$$

$$(\mid\!\sim^+)\ \frac{\Gamma, u : A \mid\!\sim B}{\Gamma, u : A \mid\!\sim B, x : \neg A \qquad \Gamma, u : A \mid\!\sim B, x : \neg\Box\neg A \qquad \Gamma, u : A \mid\!\sim B, x : B}$$

$$(\mid\!\sim^-)\ \frac{\Gamma, u : \neg(A \mid\!\sim B)}{\Gamma, x : A, x : \Box\neg A, x : \neg B}\ x \text{ new} \qquad (\Box^-)\ \frac{\Gamma, x : \neg\Box\neg A}{\Gamma, y < x, \Gamma^M_{x \rightarrow y}, y : A, y : \Box\neg A}\ y \text{ new}$$

$$(<)\ \frac{\Gamma, x < y}{\Gamma, x < y, z < y, \Gamma^M_{y \rightarrow z} \qquad \Gamma, x < y, x < z, \Gamma^M_{z \rightarrow x}} \quad \begin{array}{l} z \text{ occurs in } \Gamma \text{ and} \\ \{x < z, z < y\} \cap \Gamma = \emptyset \end{array}$$

Figure 5.12: The calculus $\mathcal{T}\mathbf{R}$. To save space, rules for → and ∨ are omitted.

simpler, intuitive and natural tableau calculus. On the other hand, even if we use labels, we do not run into problems with complexity and termination, so that we are able to define an optimal decision procedure for **R**.

The calculus makes use of labels to represent possible worlds. We consider a language $\mathcal{L}_R$ and a denumerable alphabet of labels $\mathcal{A}$, whose elements are denoted by x, y, z, ... $\mathcal{L}_R$ extends $\mathcal{L}$ by formulas of the form $\Box\neg A$ as for the other logics.

Our tableau calculus includes two kinds of labelled formulas:

- *world formulas* $x : F$, whose meaning is that F holds in the possible world represented by x;
- *relation formulas* of the form $x < y$, where $x, y \in \mathcal{A}$, used to represent the relation $<$.

We denote by $\alpha, \beta \ldots$ a world formula or a relation formula.
We define:

$$\Gamma^M_{x \rightarrow y} = \{y : \neg A, y : \Box\neg A \mid x : \Box\neg A \in \Gamma\}$$

The calculus $\mathcal{T}\mathbf{R}$ is presented in Figure 5.12. As for **P**, the rules $(\mid\!\sim^-)$ and $(\Box^-)$ that introduce new labels in their conclusion are called *dynamic rules*; all the other rules are called *static rules.*

Definition 5.46 (Truth conditions of formulas of $\mathcal{T}\mathbf{R}$)**.** *Given a model $\mathcal{M} = \langle\mathcal{W}, <, V\rangle$ and a labelled alphabet $\mathcal{A}$, we consider a mapping $I : \mathcal{A} \mapsto \mathcal{W}$. Given a formula α of the calculus $\mathcal{T}\mathbf{R}$, we define $\mathcal{M} \models_I \alpha$ as follows:*

- $\mathcal{M} \models_I x : F$ *iff* $\mathcal{M}, I(x) \models F$
- $\mathcal{M} \models_I x < y$ *iff* $I(x) < I(y)$.

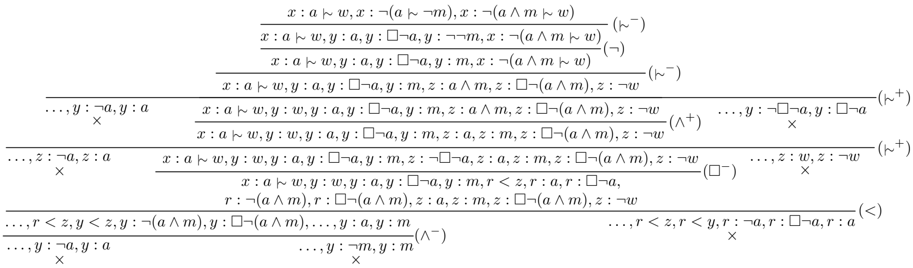

$x : a \mid\!\sim w, x : \neg(a \mid\!\sim \neg m), x : \neg(a \wedge m \mid\!\sim w)$
$(\mid\!\sim^-)$
$x : a \mid\!\sim w, y : a, y : \Box\neg a, y : \neg\neg m, x : \neg(a \wedge m \mid\!\sim w)$
$(\neg)$
$x : a \mid\!\sim w, y : a, y : \Box\neg a, y : m, x : \neg(a \wedge m \mid\!\sim w)$
$(\mid\!\sim^-)$
$x : a \mid\!\sim w, y : a, y : \Box\neg a, y : m, z : a \wedge m, z : \Box\neg(a \wedge m), z : \neg w$
$(\mid\!\sim^+)$
$\dots, y : \neg a, y : a$
×
$x : a \mid\!\sim w, y : w, y : a, y : \Box\neg a, y : m, z : a \wedge m, z : \Box\neg(a \wedge m), z : \neg w$
$(\wedge^+)$
$x : a \mid\!\sim w, y : w, y : a, y : \Box\neg a, y : m, z : a, z : m, z : \Box\neg(a \wedge m), z : \neg w$
$\dots, y : \neg\Box\neg a, y : \Box\neg a$
×
$(\mid\!\sim^+)$
$\dots, z : \neg a, z : a$
×
$x : a \mid\!\sim w, y : w, y : a, y : \Box\neg a, y : m, z : \neg\Box\neg a, z : a, z : m, z : \Box\neg(a \wedge m), z : \neg w$
$(\Box^-)$
$x : a \mid\!\sim w, y : w, y : a, y : \Box\neg a, y : m, r < z, r : a, r : \Box\neg a,$
$r : \neg(a \wedge m), r : \Box\neg(a \wedge m), z : a, z : m, z : \Box\neg(a \wedge m), z : \neg w$
$\dots, z : w, z : \neg w$
×
$(<)$
$\dots, r < z, y < z, y : \neg(a \wedge m), y : \Box\neg(a \wedge m), \dots, y : a, y : m$
$(\wedge^-)$
$\dots, y : \neg a, y : a$
×
$\dots, y : \neg m, y : m$
×
$\dots, r < z, r < y, r : \neg a, r : \Box\neg a, r : a$
×

Figure 5.13: A derivation in $\mathcal{T}\mathbf{R}$ of $\{adult \mid\!\sim worker, \neg(adult \mid\!\sim \neg married), \neg(adult \wedge married \mid\!\sim worker)\}$. To save space, we use a for $adult$, m for $married$, and w for $worker$.

We say that a set of formulas Γ *is satisfiable if, for all formulas* $\alpha \in \Gamma$*, we have that* $\mathcal{M} \models_I \alpha$*, for some model* $\mathcal{M}$ *and some mapping* I*.*

In order to verify that a set of formulas Γ is unsatisfiable, we label all the formulas in Γ with a new label x, and verify that the resulting set of labelled formulas has a closed tableau. For instance, in order to verify that the set $\{adult \mid\!\sim worker, \neg(adult \mid\!\sim \neg married), \neg(adult \wedge married \mid\!\sim worker)\}$ is unsatisfiable (and thus $adult \wedge married \mid\!\sim worker$ is entailed by $\{adult \mid\!\sim worker, \neg(adult \mid\!\sim \neg married)\}$), we can build the closed tableau in Figure 5.13.

5.5.1 Soundness, Termination, and Completeness of $\mathcal{T}\mathrm{R}$

In this section we prove that the calculus $\mathcal{T}\mathbf{R}$ is sound and complete with respect to the semantics. Moreover, we prove that, with a restriction on $(\mid\!\sim^+)$ based on the same idea of the one adopted for the other calculi, the calculus guarantees termination, that is to say, every tableau built by the calculus is finite.

First of all, we reformulate the calculus, obtaining a terminating calculus $\mathcal{T}\mathbf{R}^\mathbf{T}$. We will show in Theorem 5.56 below that, as for the calculi for $\mathbf{P}$, $\mathbf{CL}$, and $\mathbf{C}$, non-termination of the procedure due to the generation of infinitely-many worlds (thus creating infinite branches) cannot occur. As a consequence, we will observe that only a finite number of labels are introduced in a tableau (Corollary 5.57).

Similarly to the other cases, $\mathcal{T}\mathbf{R}$ does not ensure a terminating proof search due to $(\mid\!\sim^+)$, which can be applied without any control. We ensure the termination by putting on $(\mid\!\sim^+)$ in $\mathcal{T}\mathbf{R}$ the same constraint used in the other calculi. More precisely, it is easy to observe that it is useless to apply the rule on the same conditional formula more than once by using the same label x. By the invertibility of the rules (Theorem 5.49) we can assume, without loss of generality, that two applications of $(\mid\!\sim^+)$ on x are consecutive. We observe that the second application is useless, since each of the conclusions has already been obtained after the first application, and can be removed. We prevent redundant applications of $(\mid\!\sim^+)$ by keeping track of labels (worlds) in which a conditional $u : A \mid\!\sim B$ has already been applied in the current branch. To this purpose, we add to each positive conditional a list of *used* labels; we then restrict the application of $(\mid\!\sim^+)$ only to labels not occurring in the corresponding list. Notice that also the rule $(<)$ copies its principal formula $x < y$ in the conclusion; however, this rule will be applied only a finite number of times. This is a consequence

$$(\mathbf{AX})\ \Gamma, x : P, x : \neg P \quad \text{with } P \in ATM \qquad\qquad (\mathbf{AX})\ \Gamma, x < y, y < x$$

$$(\wedge^+)\frac{\Gamma, x : F \wedge G}{\Gamma, x : F, x : G} \qquad (\wedge^-)\frac{\Gamma, x : \neg(F \wedge G)}{\Gamma, x : \neg F \qquad \Gamma, x : \neg G} \qquad (\neg)\frac{\Gamma, x : \neg\neg F}{\Gamma, x : F}$$

$$(\mid\!\sim^+)\frac{\Gamma, u : A \mid\!\sim B^L}{\Gamma, x : \neg A, u : A \mid\!\sim B^{L,x} \qquad \Gamma, x : \neg\Box\neg A, u : A \mid\!\sim B^{L,x} \qquad \Gamma, x : B, u : A \mid\!\sim B^{L,x}} \ \begin{array}{l}x \text{ occurs in } \Gamma\\ \text{and } x \notin L\end{array}$$

$$(\mid\!\sim^-)\frac{\Gamma, u : \neg(A \mid\!\sim B)}{\Gamma, x : A, x : \Box\neg A, x : \neg B}\ x \text{ new} \qquad (\Box^-)\frac{\Gamma, x : \neg\Box\neg A}{\Gamma, y < x, \Gamma^M_{x \to y}, y : A, y : \Box\neg A}\ y \text{ new}$$

$$(<)\frac{\Gamma, x < y}{\Gamma, x < y, x < z, \Gamma^M_{z \to x} \qquad \Gamma, x < y, z < y, \Gamma^M_{y \to z}} \ \begin{array}{l}z \text{ occurs in } \Gamma \text{ and}\\ \{x < z, z < y\} \cap \Gamma = \emptyset\end{array}$$

Figure 5.14: The calculus $\mathcal{TR}^{\mathbf{T}}$. To save space, rules for → and ∨ are omitted.

of the side condition of the rule application and the fact that the number of labels in a tableau is finite.

The terminating calculus $\mathcal{TR}^{\mathbf{T}}$ is presented in Figure 5.14.
It is easy to prove the following structural properties of $\mathcal{TR}^{\mathbf{T}}$:

Lemma 5.47. *For any set of formulas* Γ *and any world formula* $x : F$*, there is a closed tableau for* $\Gamma, x : F, x : \neg F$*.*

Proof. By induction on the complexity of the formula F. The proof is easy and left to the reader.

Lemma 5.48 (Height-preserving admissibility of weakening)**.** *Given any set of formulas* Γ *and any formula* α*, if* Γ *has a closed tableau of height* h *then* Γ, α *has a closed tableau whose height is no greater than* h*.*

Proof. By induction on the height of the closed tableau for Γ. The proof is easy and left to the reader.

Moreover, one can easily prove that all the rules of $\mathcal{TR}^{\mathbf{T}}$ are height-preserving invertible, that is to say:

Theorem 5.49 (Height-preserving invertibility of the rules of $\mathcal{TR}^{\mathbf{T}}$)**.** *Given any rule* (**R**) *of* $\mathcal{TR}^{\mathbf{T}}$*, whose premise is* Γ *and whose conclusions are* Γ_i*, with* $i \leq 3$*, if* Γ *has a closed tableau of height* h*, then there is a closed tableau, of height no greater than* h*, for each* Γ_i*, i.e. the rules of* $\mathcal{TR}^{\mathbf{T}}$ *are height-preserving invertible.*

Proof. We consider each rule of the calculus, then we proceed by induction on the height of the closed tableau for the premise.

- $(\mid\!\sim^+)$: given a closed tableau for $\Gamma, u : A \mid\!\sim B$, then we can immediately conclude that there is also a closed tableau for $\Gamma, u : A \mid\!\sim B, x : \neg A$, for $\Gamma, u : A \mid\!\sim B, x : \neg\Box\neg A$, and for $\Gamma, u : A \mid\!\sim B, x : B$, since weakening is height-preserving admissible (see Lemma 5.48);

- $(<)$: as in the previous case, a closed tableau for the two conclusions of the rule can be obtained by weakening from the premise;
- $(\Box^-)$: given a closed tableau for $(1)\Gamma, x : \neg\Box\neg A$ and a label y not occurring in Γ, we have to show that there is a closed tableau for $\Gamma, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A$. By induction on the height of the proof of (1), we distinguish the following cases:
  - the first rule applied is $(\Box^-)$ on $x : \neg\Box\neg A$: in this case, we are done, since we have a closed tableau for $\Gamma, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A$;
  - otherwise, i.e. another rule $(\mathbf{R})$ of $\mathcal{TR}^{\mathbf{T}}$ is applied to $\Gamma, x : \neg\Box\neg A$, we can apply the inductive hypothesis on the conclusion(s) of $(\mathbf{R})$, since no rule removes side formulas in a rule application. In detail, we have that $x : \neg\Box\neg A$ belongs to all the conclusions, then we can apply the inductive hypothesis and then conclude by re-applying $(\mathbf{R})$. As an example, suppose the derivation starts with an application of $(\mid\!\sim^-)$ as follows:

  $$\frac{(1)\Gamma', u : C \mid\!\sim D, x : \neg\Box\neg A}{(2)\Gamma', v : C, v : \Box\neg C, v : \neg D, x : \neg\Box\neg A}\,(\mid\!\sim^-)$$

  We can apply the inductive hypothesis on the closed tableau for (2), concluding that there is a closed tableau for $(2')\Gamma', v : C, v : \Box\neg C, v : \neg D, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A$, from which we can conclude obtaining the following closed tableau:

  $$\frac{\Gamma', u : C \mid\!\sim D, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A}{(2')\Gamma', v : C, v : \Box\neg C, v : \neg D, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A}\,(\mid\!\sim^-)$$

  Notice that the proof has (at most) the same height of the closed tableau for (1).
- other rules: the proof is similar to the one for $(\Box^-)$ and then left to the reader.

■

Since all the rules are invertible, we have that in $\mathcal{TR}^{\mathbf{T}}$ the order of application of the rules is not relevant. Hence, no backtracking is required in the calculus, and we can assume without loss of generality that a given set of formulas Γ has a unique tableau.

Let us now prove that $\mathcal{TR}^{\mathbf{T}}$ is sound.

Theorem 5.50 (Soundness)**.** *$\mathcal{TR}^{\mathbf{T}}$ is sound with respect to rational models, i.e. if there is a closed tableau for a set of formulas Γ, then Γ is unsatisfiable.*

Proof. By induction on the height of the closed tableau for Γ. Let Γ be an axiom: if (i) $x : P \in \Gamma$ and $x : \neg P \in \Gamma$, then obviously there is no I such that, given a model $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$, we have $I(x) \in \mathcal{W}$ and $\mathcal{M}, I(x) \models P$ and $\mathcal{M}, I(x) \not\models P$. If (ii) $x < y \in \Gamma$ and $y < x \in \Gamma$, suppose on the contrary that Γ be satisfiable. Then there is a rational model $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$ and a mapping I such that $\mathcal{M} \models_I x < y$ and $\mathcal{M} \models_I y < x$. Thus we have $I(x) < I(y)$ and $I(y) < I(x)$ against the fact that $<$ is irreflexive and transitive.

For the inductive step, we prove as usual the contrapositive, that is to say, we prove for each rule that, if the premise is satisfiable, so is (at least) one of its conclusions. We only present the case of $(\Box^-)$. Since the premise is satisfiable, then there is a model $\mathcal{M}$ and a mapping I such that $\mathcal{M} \models_I \Gamma, x : \neg\Box\neg A$. Let $w \in \mathcal{W}$ such that $I(x) = w$; this means that $\mathcal{M}, w \not\models \Box\neg A$, hence there exists a world $w' < w$ such that $\mathcal{M}, w' \models A$. By the strong smoothness condition, we have that there exists a *minimal* such world, so we can assume that $w' \in Min_<(A)$, thus $\mathcal{M}, w' \models \Box\neg A$. In order to prove that the conclusion of the rule is satisfiable, we construct a mapping I' as follows: let y be a new label, not occurring in the current branch; we define (1) $I'(u) = I(u)$ for all $u \neq y$ and (2) $I'(y) = w'$. Since y does not occur in Γ, it follows that $\mathcal{M} \models_{I'} \Gamma$. By Definition 5.46, we have that $\mathcal{M} \models_{I'} y < x$ since $w' < w$. Moreover, since $I'(y) = w'$, we have that $\mathcal{M} \models_{I'} y : A$ and $\mathcal{M} \models_{I'} y : \Box\neg A$. Finally, $\mathcal{M} \models_{I'} \Gamma^M_{x \rightarrow y}$ follows from the fact that $I'(y) < I'(x)$ and from the transitivity of $<$. The single conclusion of the rule is then satisfiable in $\mathcal{M}$ via I'.

■

In order to prove the completeness of the calculus, similarly to what done for the other logics, we introduce the notion of saturated branch and we show that $\mathcal{TR}^{\mathbf{T}}$ ensures a terminating proof search. As a consequence, we will observe that the calculus introduces a finite number of labels in a tableau, and this result will be used to prove the completeness of the calculus.

Definition 5.51 (Saturated branch)**.** *We say that a branch $\mathbf{B} = \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$ of a tableau is* saturated *if the following conditions hold:*

1. *for the boolean connectives, the condition of saturation is defined in the usual way. For instance, if $x : A \wedge B \in \Gamma_i$ in $\mathbf{B}$, then there exists Γ_j in $\mathbf{B}$ such that $x : A \in \Gamma_j$ and $x : B \in \Gamma_j$;*
2. *if $x : A \mid\!\sim B \in \Gamma_i$, then for any label y in $\mathbf{B}$, there exists Γ_j in $\mathbf{B}$ such that either $y : \neg A \in \Gamma_j$ or $y : \neg\Box\neg A \in \Gamma_j$ or $y : B \in \Gamma_j$.*
3. *if $x : \neg(A \mid\!\sim B) \in \Gamma_i$, then there is a Γ_j in $\mathbf{B}$ such that, for some y, $y : A \in \Gamma_j$, $y : \Box\neg A \in \Gamma_j$, and $y : \neg B \in \Gamma_j$.*
4. *if $x : \neg\Box\neg A \in \Gamma_i$, then there exists Γ_j in $\mathbf{B}$ such that, for some y, $y < x \in \Gamma_j$, $y : A \in \Gamma_j$ and $y : \Box\neg A \in \Gamma_j$.*
5. *if $x < y \in \Gamma_i$, then for all labels z in $\mathbf{B}$, there exists Γ_j in $\mathbf{B}$ such that either $z < y \in \Gamma_j$ or $x < z \in \Gamma_j$.*

We say that a set of formulas Γ is *closed* if it is an instance of (**AX**). A branch $\mathbf{B} = \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$ is *closed* if it contains some Γ_i which is closed. A branch is *open* if it is not closed. We can prove the following Lemma:

Lemma 5.52. *Given a tableau starting with $x_0 : F$, for any open, saturated branch $\mathbf{B} = \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$, we have that:*

1. *if $z < y \in \Gamma_i$ in $\mathbf{B}$ and $y < x \in \Gamma_j$ in $\mathbf{B}$, then there exists Γ_k in $\mathbf{B}$ such that $z < x \in \Gamma_k$;*
2. *if $x : \Box\neg A \in \Gamma_i$ in $\mathbf{B}$ and $y < x \in \Gamma_j$ in $\mathbf{B}$, then there exists Γ_k in $\mathbf{B}$ such that $y : \neg A \in \Gamma_k$ and $y : \Box\neg A \in \Gamma_k$;*
3. *for no Γ_i in $\mathbf{B}$, $x < x \in \Gamma_i$.*

Proof. Let us consider an open, saturated branch $\mathbf{B} = \Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$. We consider the three claims of the lemma separately:

1. We are considering the case when $z < y \in \Gamma_i$ in $\mathbf{B}$ and $y < x \in \Gamma_j$ in $\mathbf{B}$. Since $\mathbf{B}$ is saturated, as $z < y \in \Gamma_i$, there exists Γ_k in $\mathbf{B}$ such that either $z < x \in \Gamma_k$ or $x < y \in \Gamma_k$. If $x < y \in \Gamma_k$, then the branch is closed (Γ_k is an instance of **(AX)**). Thus, we conclude that $z < x \in \Gamma_k$.

2. A relation formula $y < x$ can only be introduced by an application of either $(\Box^-)$ or $(<)$; in both cases, $\Gamma^M_{x \to y}$ is added to the current branch of the tableau. Consider any $x : \Box\neg A \in \Gamma_i$; if $j > i$, i.e. $y < x$ is introduced in the branch *after* $x : \Box\neg A$, then we are done, since $y : \neg A \in \Gamma^M_{x \to y}$ and $y : \Box\neg A \in \Gamma^M_{x \to y}$. Otherwise, if $y < x$ is introduced in the branch *before* $x : \Box\neg A$, then we are considering the case such that $x : \Box\neg A$ is introduced by an application of $(<)$, i.e. $x : \Box\neg A \in \Gamma^M_{k \to x}$ (by the presence of some $k : \Box\neg A$ in the branch) for some k and $x < k$ is also introduced in the branch. Since the branch is saturated, then either $(*)$ $x < y$ or $(**)$ $y < k$ is introduced in the branch: $(*)$ cannot be, otherwise the branch would be closed. If $(**)$ is introduced after $k : \Box\neg A$, then we are done since $y : \neg A \in \Gamma^M_{k \to y}$ and $y : \Box\neg A \in \Gamma^M_{k \to y}$; otherwise, $(**)$ has also been introduced by an application of $(<)$, and we can repeat the argument. This process must terminate. Indeed, we can observe the following facts: - a boxed formula $u : \Box\neg A$ must be initially introduced in a branch by an application of either $(\Box^-)$ or $(\mid\!\sim^-)$, further applications of $(<)$ and $(\Box^-)$ may only "propagate" it to other worlds. In both cases, $(\Box^-)$ and $(\mid\!\sim^-)$, u is a new label not occurring in the branch, so that all formulas $v < u$ will necessarily be introduced when $u : \Box\neg A$ already belongs to the branch. Let therefore u be a label such that $y < u$ is introduced in the branch *after* $u : \Box\neg A$; by saturation we have that $y : \neg A$ and $y : \Box\neg A$ belong to the branch.

3. A relation $x < x$ cannot be introduced by rule $(\Box^-)$, since this rule establishes a relation between x in $\mathbf{B}$ and a label distinct from x. On the other hand, it cannot be introduced by modularity. Indeed, for rule $(<)$ to introduce a relation $x < x$, there must be in $\mathbf{B}$ some relation $y < x$ (resp. $x < y$) for some y. But in this case the side condition of the rule would not be fulfilled, and the rule could not be applied.

■

Also in $\mathcal{T}\mathbf{R}^\mathbf{T}$ we introduce the restriction on the order of application of the rules, as adopted for the other systems (see Definition 5.13). That is to say: the application of the $(\Box^-)$ rule is postponed to the application of all propositional rules and to the test of whether Γ is an instance of (**AX**) or not.

Similarly to the case of $\mathcal{T}\mathbf{P}^\mathbf{T}$ and $\mathcal{T}\mathbf{CL}^\mathbf{T}$, we can show that $\mathcal{T}\mathbf{R}^\mathbf{T}$ ensures termination. In particular, the rule $(\mid\!\sim^-)$ can be applied only once for each negated conditional Γ. Furthermore, the generation of infinite branches due to the interplay between rules $(\mid\!\sim^+)$ and $(\Box^-)$ cannot occur. Indeed, each application of $(\Box^-)$ to a formula $x : \neg\Box\neg A$ (introduced by $(\mid\!\sim^+)$) adds the formula $y : \Box\neg A$ to the conclusion, so that $(\mid\!\sim^+)$ can no longer consistently introduce $y : \neg\Box\neg A$. In order to prove this result in a rigorous manner, we proceed as follows: first, we introduce the complexity measure on a set of formulas Γ of Definition 5.54, denoted by $m(\Gamma)$, which consists of five measures c_1, c_2, c_3, c_4 and c_5 in a lexicographic order, and the auxiliary Definition 5.53; then, we prove that each application of $\mathcal{T}\mathbf{R}^\mathbf{T}$'s rules reduces this measure, until the rules are no longer applicable, or leads to a closed tableau.

The complexity of a formula $cp(F)$ is defined in Definition 5.17; moreover, we use square brackets $[\ldots]$ to denote *multisets*.

Definition 5.53. *Given an initial set of formulas* Γ_0*, we define:*

- *the set* $\mathcal{L}^{\Gamma_0}_{\Box+}$ *of boxed formulas* $\Box\neg A$ *that can be generated in a tableau for* Γ_0*, i.e.* $\mathcal{L}^{\Gamma_0}_{\Box+} = \{\Box\neg A \mid A \mid\!\sim B \in_+ \Gamma_0\} \cup \{\Box\neg A \mid A \mid\!\sim B \in_- \Gamma_0\}$*. We let* $n_0 =\mid \mathcal{L}^{\Gamma_0}_{\Box+} \mid$*;*
- *the* multiset $\mathcal{L}^{\Gamma_0}_{\Box-}$ *of negated boxed formulas that can be generated in a tableau for* Γ_0*, i.e.* $\mathcal{L}^{\Gamma_0}_{\Box-} = [\neg\Box\neg A \mid A \mid\!\sim B \in_+ \Gamma_0]$*. We let* $k_0 =\mid \mathcal{L}^{\Gamma_0}_{\Box-} \mid$*;*

Given a label x *and a set of formulas* Γ *in the tableau for the initial set* Γ_0*, we define:*

- *the number* n_x *of positive boxed formulas* $\Box\neg A$ *not labelled by* x*, i.e.* $n_x = n_0 - \mid \{\Box\neg A \in \mathcal{L}^{\Gamma_0}_{\Box+} \mid x : \Box\neg A \in \Gamma\} \mid$*;*
- *the number* k_x *of negated boxed formulas* $\neg\Box\neg A$ *not yet expanded in a world* x*, i.e.* $k_x = k_0 - \mid [\neg\Box\neg A \in \mathcal{L}^{\Gamma_0}_{\Box-} \mid y : \Box\neg A \in \Gamma$ *and* $y < x \in \Gamma] \mid$[4]*.*

Definition 5.54 (Lexicographic order)**.** *We define* $m(\Gamma) = \langle c_1, c_2, c_3, c_4, c_5 \rangle$ *where:*

- $c_1 =\mid \{u : A \mid\!\sim B \in_- \Gamma\} \mid$
- c_2 *is the multiset given by* $[c_2^{x_1}, c_2^{x_2}, \ldots, c_2^{x_n}]$*, where* $x_1, x_2, \ldots, x_n$ *are the labels occurring in* Γ *and, given a label* x*,* c_2^x *is a pair* (n_x, k_x) *in a lexicographic order (*n_x *and* k_x *are defined as in Definition 5.53). We consider the integer multiset ordering given by* c_2
- $c_3 =\mid \{\langle x, A \mid\!\sim B \rangle \mid u : A \mid\!\sim B^L \in \Gamma$ *and* $x \notin L\} \mid$
- $c_4 = \sum_z c_4^z$*, where* z *occurs in* Γ *and* $c_4^z =\mid \{x < y \in \Gamma \mid \{x < z, z < y\} \cap \Gamma = \emptyset\} \mid$
- $c_5 = \sum_{x:F \in \Gamma} cp(F)$

We consider the lexicographic order given by $m(\Gamma)$*.*

Roughly speaking, c_1 is the number of negated conditionals that can still be expanded in the tableau. The application of ($\mid\!\sim^-$) reduces c_1. c_2 keeps track of positive conditionals *which can still create a new world*. The application of ($\Box^-$) reduces c_2. c_3 represents the number of positive conditionals not yet expanded in a world x: the application of ($\mid\!\sim^+$) reduces this measure, since the rule is applied to $u : A \mid\!\sim B^L$ by using x *only if* x does not belong to L, i.e. $u : A \mid\!\sim B$ has not yet been expanded in x. c_4 represents the number of relations $x < y$ not yet added to the current branch: the application of ($<$) reduces c_4, since it applies the modularity of $<$ in case $x < y$ and, given z, neither $z < y$ nor $x < z$ belong to the current set of formulas. c_5 is the sum of the complexities of the world formulas in Γ: an application of the rules for boolean connectives reduces c_5.

First of all, we prove that the application of any rule of $\mathcal{TR}^{\mathbf{T}}$ reduces $m(\Gamma)$ or leads to a closed tableau, as stated by the following Lemma:

Lemma 5.55. *Let* Γ' *be a set of formulas obtained as a conclusion of an application of a rule of* $\mathcal{TR}^{\mathbf{T}}$ *to a set of formulas* Γ*. We have that either the tableau for* Γ' *is closed or* $m(\Gamma') < m(\Gamma)$*.*

[4] Notice that, in case there are two positive conditionals $A \mid\!\sim B$ and $A \mid\!\sim C$ with the same antecedent, then the multiset $\mathcal{L}^{\Gamma_0}_{\Box-}$ contains two instances of $\neg\Box\neg A$. Therefore, if the rule ($\Box^-$) is applied to $x : \neg\Box\neg A$ (for instance, generated by an application of ($\mid\!\sim^+$) in x on $A \mid\!\sim B$), then k_x decreases only by 1 unit, whereas the second instance of $\neg\Box\neg A$, i.e. the one "associated" with $A \mid\!\sim C$, is still considered to be *not expanded* in x, thus it still "contributes" to k_x.

Proof. We consider each rule of the calculus:

- $(\mid\!\sim^-)$: an application of this rule reduces c_1, since it is applied to a negated conditional $u : \neg(A \mid\!\sim B)$ belonging to its premise which is removed from the conclusion;
- $(\Box^-)$: first of all, observe that c_1 is not augmented in the conclusion, since no negated conditional is added by the rule. The application of $(\Box^-)$ reduces c_2 or leads to a closed tableau. We are considering the following rule application:

$$\frac{\Gamma, x : \neg\Box\neg A}{\Gamma, y < x, \Gamma^M_{x\rightarrow y}, y : A, y : \Box\neg A}\ (\Box^-)$$

  where y is a new label. c_2 in the premise, say c_{2_p}, is a multiset $[\dots, c^x_{2_p}, \dots]$, whereas in the conclusion we have to consider a measure, called c_{2_c}, of the form $[\dots, c^x_{2_c}, c^y_{2_c}, \dots]$. By the standard definition of integer multiset ordering, we prove that either $c_{2_c} < c_{2_p}$ (by showing that $c^x_{2_c} < c^x_{2_p}$ and $c^y_{2_c} < c^x_{2_p}$, i.e. we replace an integer $c^x_{2_p}$ with two smaller integers, see [DM79] for details on integer multiset orderings) or that the procedure leads to a closed tableau. We conclude the proof as follows:

  - let us consider $c^y_{2_c}$ and $c^x_{2_p}$. By definition, $c^y_{2_c}$ is a pair (n_{y_c}, k_{y_c}) and $c^x_{2_p}$ is a pair (n_{x_p}, k_{x_p}). We distinguish two cases: if $x : \Box\neg A \notin \Gamma$, then we easily prove that $n_{y_c} < n_{x_p}$, since $y : \Box\neg A$ belongs to the conclusion: therefore, the number of boxed formulas $\Box\neg A$ not occurring with label y is smaller than the number of boxed formulas not occurring with label x, and we are done (remember that all the positive boxed formulas labelled by x are also labelled by y in the conclusion, by the presence of $\Gamma^M_{x\rightarrow y}$); if $x : \Box\neg A \in \Gamma$, then the application of $(\Box^-)$ leads to a node containing both $y : A$ and $y : \neg A$ and, by the restriction on the order of application of the rules (see Definition 5.13), the procedure terminates building a closed tableau;
  - let us consider $c^x_{2_c}$ and $c^x_{2_p}$. It is easy to observe that $n_{x_p} = n_{x_c}$, since no formula $x : \Box\neg A$ is added nor removed in the conclusion (the positive boxed formulas labelled by x are the same in both the premise and the conclusion). We conclude since $k_{x_c} < k_{x_p}$, since $x : \neg\Box\neg A$ has been expanded in x, so it "contributes" to k_{x_p} whereas it does not to k_{x_c}.

- $(\mid\!\sim^+)$: first of all, notice that c_1 and c_2 cannot be higher in the conclusions than in the premise. Notice that the addiction of $x : \neg\Box\neg A$ in the inner conclusion does not increase c_2, since $\neg\Box\neg A$ is a negated boxed formula still to be considered in both the premise and the conclusions. The application of $(\mid\!\sim^+)$ reduces c_3. Suppose that this rule is applied to $u : A \mid\!\sim B^L$ by using label x in the conclusions; by the restriction in Figure 5.14, this means that $x \notin L$, so $\langle x, A \mid\!\sim B\rangle$ belongs to the set whose cardinality determines c_3 in the premise. Obviously, since x is added to L for $u : A \mid\!\sim B$ in the three conclusions of the rule, we can easily observe that $\langle x, A \mid\!\sim B\rangle$ does no longer belong to the set in the definition of c_3 in the conclusions: c_3 is then smaller in the conclusions than in the premise, and we are done;
- $(<)$: the application of $(<)$ reduces c_4, whereas c_1, c_2, and c_3 cannot be augmented (at most formulas $z : \Box\neg A$ are added in the conclusions by $\Gamma^M_{z\rightarrow x}$ and $\Gamma^M_{y\rightarrow z}$, reducing c_2). To conclude the proof, just observe that, given a label z

and a formula $x < y$, $(<)$ is applied if $\{x < z, z < y\} \cap \Gamma = \emptyset$, i.e. $x < y$ belongs to the set used to define c_4^z in the premise, say $c_{4_p}^z$. When the rule is applied, in the left premise $z < y$ is added, and $x < y$ does no longer belong to the set used to define c_4^z in the conclusion, say $c_{4_c}^z$. Therefore, $c_{4_c}^z < c_{4_p}^z$. The same for the right premise, and we are done;

- rules for the boolean connectives: these rules do not increase values of c_1, c_2, c_3, and c_4. Their application reduces c_5, since the (sum of) complexity of the subformula(s) introduced in the conclusion(s) is lower then the complexity of the principal formula to which the rule is applied.

■

Now we can prove that $\mathcal{TR}^{\mathbf{T}}$ ensures a terminating proof search:

Theorem 5.56 (Termination of $\mathcal{TR}^{\mathbf{T}}$)**.** *Let Γ be a finite set of formulas, then any tableau generated by $\mathcal{TR}^{\mathbf{T}}$ is finite.*

Proof. Let Γ' be obtained by an application of a rule of $\mathcal{TR}^{\mathbf{T}}$ to a premise Γ. By Lemma 5.55 we have that either the procedure leads to a closed tableau (and in this case we are done) or we have that $m(\Gamma') < m(\Gamma)$. This means that, similarly to the case of $\mathbf{P}$, a finite number of applications of the rules leads either to close the branch or to a node whose measure is as follows: $\langle 0, [(0,0),(0,0),\ldots,(0,0)], 0, 0, c_{5_{min}}\rangle$, where $c_{5_{min}}$ is the minimal value that c_5 can assume for Γ. This means that no rule of $\mathcal{TR}^{\mathbf{T}}$ is further applicable. This is a consequence of the following facts:

- $(\mid\!\sim^-)$ is no longer applicable, since $c_1 = 0$;
- since $c_2 = [(0,0),(0,0),\ldots,(0,0)]$, given any label x, we have that $c_2^x = (0,0)$. Therefore, the $(\Box^-)$ rule is no longer applicable, since we can observe that there is no $x : \neg\Box\neg A \in \Gamma$ (by the fact that $k_x = 0$);
- the rule $(\mid\!\sim^+)$ is not further applicable since $c_3 = 0$; indeed, $c_3 = 0$ means that all formulas $A \mid\!\sim B$ have already been expanded in each world x;
- $c_4 = 0$, i.e. for all formulas $x < y$, given any label z, we have that either $z < y \in \Gamma$ or $x < z \in \Gamma$, thus the $(<)$ rule is not further applicable;
- since c_5 assumes its *minimal* value $c_{5_{min}}$, no rule for a boolean connective is further applicable. If a boolean rule is applicable, then its application reduces the value of c_5 in its conclusion(s) by Lemma 5.55, against the minimality of $c_{5_{min}}$in the premise.

■

As a consequence of Theorem 5.56, we can observe that the tableau for a given set of formulas Γ contains a finite number of labels, since all the branches in a tableau generated by $\mathcal{TR}^{\mathbf{T}}$ are finite.

Corollary 5.57. *Given a set of formulas Γ, the tableau generated by $\mathcal{TR}^{\mathbf{T}}$ for Γ only contains a finite number of labels.*

Let us now show that $\mathcal{TR}^{\mathbf{T}}$ is complete with respect to the semantics:

Theorem 5.58 (Completeness)**.** *$\mathcal{TR}^{\mathbf{T}}$ is complete with respect to rational models, i.e. if a set of formulas Γ is unsatisfiable, then it has a closed tableau in $\mathcal{TR}^{\mathbf{T}}$.*

Proof. We show the contrapositive, i.e. if there is no closed tableau for Γ, then Γ is satisfiable. Consider the tableau starting with the set of formulas $\{x : F$ such that $F \in \Gamma\}$ and any open, saturated branch $\mathbf{B} = \Gamma_1, \Gamma_2, \ldots, \Gamma_n$ in it. Starting from $\mathbf{B}$, we build a canonical model $\mathcal{M} = \langle \mathcal{W}_B, <, V\rangle$ satisfying Γ, where:

- $\mathcal{W}_B$ is the set of labels that appear in the branch $\mathbf{B}$;
- for each $x, y \in \mathcal{W}_B$, $x < y$ iff there exists Γ_i in $\mathbf{B}$ such that $x < y \in \Gamma_i$;
- for each $x \in \mathcal{W}_B$, $V(x) = \{P \in ATM \mid$ there is Γ_i in $\mathbf{B}$ such that $x : P \in \Gamma_i\}$.

We can easily prove that:
(*i*) by Corollary 5.57, we have that $\mathcal{W}_B$ is finite;
(*ii*) $<$ is an irreflexive, transitive and modular relation on $\mathcal{W}_B$ satisfying the smoothness condition. Irreflexivity, transitivity and modularity are obvious, given Definition 5.51 and Lemma 5.52 above. Since $<$ is irreflexive and transitive, it can be easily shown that it is also acyclic. This property together with the finiteness of $\mathcal{W}_B$ entails that $<$ cannot have infinite descending chains. In turn this last property together with the transitivity of $<$ entails the smoothness condition.
(*iii*) We show that, for all formulas F and for all Γ_i in $\mathbf{B}$, (i) if $x : F \in \Gamma_i$ then $\mathcal{M}, x \models F$ and (ii) if $x : \neg F \in \Gamma_i$ then $\mathcal{M}, x \not\models F$. The proof is by induction on the complexity of the formulas. If $F \in ATM$ this immediately follows from definition of V. For the inductive step, we only present the case of $F = A \mid\sim B$. The other cases are similar and then left to the reader. Let $x : A \mid\sim B \in \Gamma_i$. By Definition 5.51, we have that, for all y, there is Γ_j in $\mathbf{B}$ such that either $y : \neg A \in \Gamma_j$ or $y : B \in \Gamma_j$ or $y : \neg\Box\neg A \in \Gamma_j$. We show that for all $y \in Min_<(A)$, $\mathcal{M}, y \models B$. Let $y \in Min_<(A)$. This entails that $\mathcal{M}, y \models A$, hence $y : \neg A \notin \Gamma_j$. Similarly, we can show that $y : \neg\Box\neg A \notin \Gamma_j$. It follows that $y : B \in \Gamma_j$, and by inductive hypothesis $\mathcal{M}, y \models B$. (ii) If $x : \neg(A \mid\sim B) \in \Gamma_i$, since $\mathbf{B}$ is saturated, there is a label y in some Γ_j such that $y : A \in \Gamma_j$, $y : \Box\neg A \in \Gamma_j$, and $y : \neg B \in \Gamma_j$. By inductive hypothesis we can easily show that $\mathcal{M}, y \models A$, $\mathcal{M}, y \models \Box\neg A$, hence $y \in Min_<(A)$, and $\mathcal{M}, y \not\models B$, hence $\mathcal{M}, x \not\models A \mid\sim B$.

Since $\mathcal{TR}^{\mathbf{T}}$ makes use of the restriction in Figure 5.14, we have to show that this restriction preserves the completeness. We have only to show that if $(\mid\sim^+)$ is applied twice on the same conditional $A \mid\sim B$, in the same branch, by using the same label x, then the second application is useless. Since all the rules are invertible (Theorem 5.49), we can assume, without loss of generality, that the two applications of $(\mid\sim^+)$ are consecutive. We conclude that the second application is useless, since each of the conclusions has already been obtained after the first application, and can be removed.

■

The construction of the model done in the proof of Theorem 5.58 provides a constructive proof of the finite model property of $\mathbf{R}$:

Corollary 5.59 (Finite model property). **R** *has the finite model property.*

5.5.2 Complexity of R

In this section we define a systematic procedure which allows the satisfiability problem for $\mathbf{R}$ to be decided in nondeterministic polynomial time, in accordance with the known complexity results for this logic.

Let n be the size of the starting set Γ of which we want to verify the satisfiability. The number of applications of the rules is proportional to the number of labels introduced in the tableau. In turn, this is $O(2^n)$ due to the interplay between the rules $(\mid\!\sim^+)$ and $(\Box^-)$. Hence, the complexity of $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ is exponential in n.

In order to obtain a better complexity bound for validity in $\mathbf{R}$ we present the following procedure. Intuitively, we do not apply $(\Box^-)$ to all negated boxed formulas, but only to formulas $y : \neg\Box\neg A$ not already expanded, i.e. such that $z : A, z : \Box\neg A$ do not belong to the current branch. As a result, we build a *small* model for the initial set of formulas (see Theorem 1 in [GGOP06a]).

Let us define a nondeterministic procedure CHECK(Γ) to decide whether a given set of formulas Γ is satisfiable. We make use of a procedure EXPAND(Γ) that returns one saturated expansion of Γ with respect to all static rules. In case of a branching rule, EXPAND nondeterministically selects (guesses) one conclusion of the rule.

```
CHECK($\Gamma$)
1. $\Gamma \longleftarrow$ EXPAND($\Gamma$);
2. if $\Gamma$ contains an axiom then return unsat;
3. for each $x : \neg(A \mid\!\sim B) \in \Gamma$ do $\Gamma \longleftarrow$APPLY($\Gamma, \mid\!\sim^-, x : \neg(A \mid\!\sim B)$);
4. $\Gamma \longleftarrow$ EXPAND($\Gamma$);
5. if $\Gamma$ contains an axiom then return unsat;
while $\Gamma$ contains a $y : \neg\Box\neg A$ not marked as CONSIDERED do
    6. select $y : \neg\Box\neg A \in \Gamma$ not already marked as CONSIDERED;
        6a. if there is $z$ in $\Gamma$ such that $z : A \in \Gamma$ and $z : \Box\neg A \in \Gamma$
            then 6a'. add $z < y$ and $\Gamma^M_{y \rightarrow z}$ to $\Gamma$;
            else 6a". $\Gamma \longleftarrow$ APPLY($\Gamma, \Box^-, y : \neg\Box\neg A$);
        6b. mark $y : \neg\Box\neg A$ as CONSIDERED;
    7. $\Gamma \longleftarrow$ EXPAND($\Gamma$);
    8. if $\Gamma$ contains an axiom then return unsat;
endWhile
9. return satisf;
```

Observe that the addition of the set of formulas $z < y, \Gamma^M_{y \rightarrow z}$ in step 6a' could be omitted and it has been added mostly to enhance the understanding of the procedure. Indeed, the rule $(<)$, which is applied at each iteration to assure modularity, already takes care of adding such formulas. The procedure CHECK nondeterministically builds an open branch for Γ.

Theorem 5.60 (Soundness and completeness of the procedure)**.** *The above procedure is sound and complete with respect to the semantics.*

Proof. (*Soundness*). We prove that if the initial set of formulas Γ is satisfiable, then the above procedure returns satisf. More precisely, we prove that each step of the procedure preserves the satisfiability of Γ. As far as EXPAND is concerned, notice that it only applies the static rules of $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ these rules preserve satisfiability (Theorem 5.50). Consider now step 6. Let $y : \neg\Box\neg A$ the formula selected in this step. If $(\Box^-)$ is applied to $y : \neg\Box\neg A$ (step 6a") we are done, since $(\Box^-)$ preserves satisfiability (Theorem 5.50). If Γ already contains $z : A, z : \Box\neg A$, then step 6a' is executed, and the relation $z < y$ is added. In this case we reason as follows. Since Γ is satisfiable, we have that there is a model $\mathcal{M}$ and a mapping I such that (1) $\mathcal{M}, I(y) \models \neg\Box\neg A$ and (2) $\mathcal{M}, I(z) \models A$ and $\mathcal{M}, I(z) \models \Box\neg A$. We can observe that $I(z) < I(y)$ in $\mathcal{M}$. Indeed, by the truth condition of $\neg\Box\neg A$ and by the strong smoothness condition, we have that there exists w such that $w < I(y)$ and $\mathcal{M}, w \models A, \Box\neg A$. By modularity of $<$, either 1. $w < I(z)$ or 2. $I(z) < I(y)$. (1) is impossible, since otherwise we would

have $\mathcal{M}, w \models \neg A$, which contradicts $\mathcal{M}, w \models A$. Hence, 2 holds. Therefore, we can conclude that step 6a' preserves satisfiability.
(*Completeness*). It can be easily shown that in case the procedure above returns `satisf`, then the branch generated by the procedure is saturated (see Definition 5.51). Therefore, we can build a model for the initial Γ as shown in the proof of Theorem 5.58.

■

Theorem 5.61 (Complexity of the `CHECK` procedure)**.** *By means of the procedure `CHECK` the satisfiability of a set of formulas of logic* **R** *can be decided in nondeterministic polynomial time.*

Proof. Observe that the procedure generates at most *O(n)* labels by applying the rule $(\mid\!\sim^-)$ (step 3) and that the while loop generates at most one new label for each $\neg\Box\neg A$ formula. Indeed, the rule $(\Box^-)$ is applied to a labelled formula $y : \neg\Box\neg A$ to generate a new world only if there is not a label z such that $z : A \in \Gamma$ and $z : \Box\neg A \in \Gamma$ are already on the branch. In essence, the procedure does not add a new minimal A-world on the branch if there is already one. As the number of different $\neg\Box\neg A$ formulas is at most $O(n)$, then the while loop can add at most $O(n)$ new labels on the branch. Moreover, for each different label x, the expansion step can add at most $O(n)$ formulas $x : \neg\Box\neg A$ on the branch, one for each positive conditional $A \mid\!\sim B$ occurring in the set Γ. We can therefore conclude that the while loop can be executed at most $O(n^2)$ times.

As the number of generated labels is at most $O(n)$, by the subformula property, the number of labelled formulas on the branch is at most $O(n^2)$. Hence, the execution of step 6a has complexity $O(n^2)$. The execution of the nondeterministic procedure `EXPAND` has complexity $O(n^2)$, including a guess of size $O(n^2)$, whereas to check if Γ is an axiom has complexity $O(n^4)$ (since it requires to check whether, for each labelled formula $x : P \in \Gamma$, the formula $x : \neg P$ is also in Γ, and Γ contains at most $O(n^2)$ labelled formulas). We can therefore conclude that the execution of the `CHECK` procedure requires at most $O(n^6)$ steps, i.e. it has a polynomial complexity.

■

Given that the problem of deciding validity for rational logic **R** is known to be **coNP**-complete [LM92], by Theorem 5.61 we can conclude that the procedure `CHECK` is optimal for **R**.

Theorem 5.62 (Complexity of **R**)**.** *The problem of deciding validity for rational logic* **R** *is* **coNP***-complete.*

5.6 Conclusions and Comparison with Other Works

In this chapter, we have presented tableau calculi for all the KLM logical systems for default reasoning, namely **R**, **P**, **CL**, and **C**. We have given a tableau calculus for preferential logic **P**, loop-cumulative logic **CL**, and cumulative logic **C**. Then we have presented a *labelled* tableau calculus for the rational logic **R**. The calculi presented give a decision procedure for the respective logics. Moreover, for **R**, **P** and **CL** we have shown that we can obtain **coNP** decision procedures by refining the rules of the respective calculi. In case of **C**, we obtain a decision procedure by adding a suitable loop-checking mechanism. Our procedure gives an hyper exponential upper bound. Further investigation is needed to get a more efficient procedure. On the other hand, we are not aware of any tighter complexity bound for this logic.

We briefly remark on some related works (for a broader discussion, see Section 1.4.2).

Artosi, Governatori, and Rotolo [AGR02] develop a labelled tableaux calculus for **C**. Their calculus is based on the interpretation of **C** as a conditional logic with a selection function semantics. As a major difference from our approach, their calculus makes use of labelled formulas, where the labels represent possible worlds or sets of possible worlds. World labels are annotated by formulas, in their turn, to express minimality assumptions (e.g. that a world w is a minimal A-world, or in terms of the selection function, belongs to $f(A, u)$, is represented by a label like w^A, to simplify their treatment). They use then a sophisticated unification mechanism on the labels to match two annotated worlds, e.g. w^A, w^B; observe that by CSO (which is equivalent to CUT+CM), the equivalence of A and B might also be enforced by the conditionals contained in a tableau branch. Even if they do not discuss decidability and complexity issues, their tableau calculus should give a decision procedure for **C**.

In [GGOS03] and [GGOS05] it is defined a labelled tableaux calculus for the logic **CE** and some of its extensions. The flat fragment of **CE** corresponds to the system **P**. The similarity between the two calculi lies in the fact that both approaches use a modal interpretation of conditionals. The major difference is that the calculus presented here does not use labels, whereas the one proposed in [GGOS03] does. A further difference is that in [GGOS03] the termination is obtained by means of a loop-checking machinery, and it is not clear if it matches complexity bounds and if it can be adapted in a simpler way to **CL** and to **C**.

Lehmann and Magidor [LM92] propose a non-deterministic algorithm that, given a finite set K of conditional assertions $C_i \mid\!\sim D_i$ and a conditional assertion $A \mid\!\sim B$, checks if $A \mid\!\sim B$ is not entailed by K in the logic **P**. This is an abstract algorithm useful for theoretical analysis, but practically unfeasible, as it requires to guess sets of indexes and propositional evaluations. They conclude that entailment in **P** is **coNP**, thus obtaining a complexity result similar to ours. However, it is not easy to compare their algorithm with our calculus, since the two approaches are radically different. As far as the complexity result is concerned, notice that our result is more general than theirs, since our language is richer: we consider boolean combinations of conditional assertions (and also combinations with propositional formulas), whereas they do not. As remarked by Boutilier [Bou94], this more general result is not an obvious consequence of the more restricted one. Moreover, we prove the **coNP** result also for the system **CL**. At the best of our knowledge, this result was unknown up to now.

Chapter 6

Theorem Provers for Conditional and KLM Logics

In this chapter we introduce CondLean, a theorem prover for conditional logics, and KLMLean, a theorem prover for KLM logics. Moreover, we introduce GoalD$\mathcal{U}$CK, an implementation of the goal directed proof procedure $\mathcal{U}$S' described in Chapter 4.

CondLean and KLMLean are both inspired to the "lean" methodology, whose basic idea is to write short programs and exploit the power of Prolog's engine as much as possible. Both our theorem provers also comprise a graphical interface written in Java.

Preliminary descriptions of CondLean have been presented in [OP03, OP05a, OP05b, OP07]. One can also find a preliminary description of KLMLean in [OP05c, GGP07]. GoalD$\mathcal{U}$CK has been introduced in [OP08]. Our theorem provers, together with their source code, are available for free download at the following addresses:

CondLean	`http://www.di.unito.it/~pozzato/condlean 3.2`
KLMLean	`http://www.di.unito.it/~pozzato/klmlean 2.0`
GoalD$\mathcal{U}$CK	`http://www.di.unito.it/~pozzato/goalduck`

6.1 Introduction

In this chapter we introduce two theorem provers:

- **CondLean**, a theorem prover for conditional logics;
- **KLMLean**, a theorem prover for KLM logics.

To the best of our knowledge, these are the first theorem provers for these logics.

Both CondLean and KLMLean are SICStus Prolog implementations of the calculi introduced in the two previous chapters, namely CondLean implements sequent calculi SeqS introduced in Chapter 4, whereas KLMLean is an implementation of the tableau calculi introduced in Chapter 5.

CondLean and KLMLean are inspired (and *only* inspired) to the so-called “lean” methodology, introduced by Beckert and Posegga in the middle of the 90s [BP95, BP96, Fit98]. Beckert and Posegga have proposed a very elegant and extremely efficient first-order theorem prover, called leanT^AP, consisting of only five Prolog clauses. The basic idea of the “lean” methodology is “to achieve maximal efficiency from minimal means” [BP95], that is to say to write short programs and exploit the power of Prolog’s engine as much as possible. In the following years, Beckert and Goré [BG97, BG01] have presented a “lean”, efficient, and modular labelled tableau calculus for all the fifteen basic propositional modal logics. More in detail, they have introduced a *free* variable “lean” tableau method, where the labels contain free and universal variables. They have also presented an implementation of this calculus, called leanK. The version for the basic modal logic K consists of just eleven Prolog clauses, whereas the version for the logic KD consists of only six clauses. The performances of leanK are promising, especially compared with other more complex implementations for modal deduction.

The core of both CondLean and KLMLean consists in a set of clauses, and each one of them represents a sequent/tableau rule or axiom; the proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism. As mentioned, it is worth noticing that CondLean and KLMLean are only inspired to the “lean” methodology, but they do not fit its style in a rigorous manner. Indeed, our theorem provers make use of some auxiliary predicates, either offered by the SICStus Prolog libraries (such as `member`) or part of our implementation (e.g. a predicate `generateLabel` used to create a new label in the proof tree). This is enough to conclude that they are *not “really-lean”* theorem provers.

CondLean and KLMLean also comprise a graphical user interface (GUI) written in Java.

In the middle of this chapter, we also briefly introduce GoalD$\mathcal{U}$CK, a very simple SICStus Prolog implementation of the goal-directed calculus $\mathcal{U}$S’ introduced in Chapter 4.

6.2 CondLean: a Theorem Prover for Conditional Logics

In this section we describe an implementation of SeqS calculi in SICStus Prolog. The program, called CondLean, supports the basic conditional system CK, its extensions with ID, MP, CS, and CEM and all their combinations, except those combining both CEM and MP, i.e. all the conditional logics for which we have described sound and complete sequent calculi (Chapter 4); as far as we know this is the first theorem prover for these logics.

For each supported conditional system, CondLean offers three different implementations, namely:

- a simple version, called *constant labels*, where Prolog *constants* are used to represent SeqS’s labels;
- a more efficient one, called *free variables*, where labels are represented by Prolog *variables*, inspired by the free-variable tableaux presented in [BG97];

- an *heuristic version*, implementing a “two-phase” theorem prover, which first attempts to prove a sequent by using an incomplete, but fast, proof procedure (phase 1), and then it calls the free-variable proof procedure (phase 2) in case of failure.

CondLean is inspired to the “lean” methodology mentioned above; in particular, the core of CondLean consists in a set of clauses, each one of them represents a sequent rule or axiom; the proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism. As mentioned, CondLean is only inspired to the “lean” methodology, but it does not fit its style in a rigorous manner; in Section 6.2.5 we present a possible “*really-lean*” implementation of SeqS calculi, and discuss how CondLean seems to be a better solution for the calculi implemented in this work.

6.2.1 The Constant Labels Version

We represent each component of a sequent (antecedent and consequent) by a *list* of formulas, partitioned into three sub-lists: atomic formulas, transitions and complex formulas. Atomic and complex formulas are represented by a list like `[x,a]`, where `x` is a Prolog constant and `a` is a formula. A transition $x \xrightarrow{A} y$ is represented by `[x,a,y]`. SeqS’s labels are represented by Prolog’s constants. The sequent calculi are implemented by the predicate

prove(Cond, Sigma, Delta, Labels).

which succeeds if and only if $\Sigma \vdash \Delta$ is derivable in SeqS, where `Sigma` and `Delta` are the lists representing the multisets Σ and Δ, respectively and `Labels` is the list of labels introduced in that branch. `Cond` is a list of pairs of kind $[F, \mathit{Used}]$, where F is a conditional formula `[X,A => B]` and *Used* is a list of transitions $[[X, A_1, Y_1], ..., [X, A_n, Y_n]]$ such that ($\Rightarrow$ L) has already been applied to $x : A \Rightarrow B$ by using transitions $x \xrightarrow{A_i} y_i$. The list `Cond` is used in order to ensure the termination of the proof search, by applying the restrictions stated by Lemmas 4.18, 4.20, 4.27, and 4.33. In fact, ($\Rightarrow$ L) is applied to $x : A \Rightarrow B$ by choosing $x \xrightarrow{A} y$ such that $x \xrightarrow{A_y} y$ belongs to `Sigma` and $[x, a_y, y]$ *does not belong to* `Used`.

For instance, to prove $x : B \vdash x : A \Rightarrow A$ in CK+ID, one queries CondLean with the goal

```
prove([],[[[x,b]],[],[]], [[],[],[[x,a=>a]]], [x]).
```

Each clause of the `prove` predicate implements one axiom or rule of SeqS. The following ones are clauses implementing axioms:

```
prove(_,[LitSigma,_,_],[LitDelta,_,_],_):-
  member(F,LitSigma),member(F,LitDelta),!.
prove(_,[LitSigma,_,_],_,_):-
  member([_,false],LitSigma),!.
```

The above clauses succeed if the same world formula F belongs to both the left hand side and the right hand side of the sequent and if a formula $x : \bot$ belongs to the left hand side of the sequent, respectively. As another example, the clause[1] implementing ($\Rightarrow$ L) is as follows:

```
prove(Cond,[LitSigma,TransSigma,ComplexSigma],
      [LitDelta,TransDelta,ComplexDelta], Labels):-
   member([X,A => B],ComplexSigma),select([[X,A => B],Used],Cond,TempCond),
   member([X,C,Y],TransSigma), \+member([X,C,Y],Used),!,
   put([Y,B],LitSigma,ComplexSigma,NewLitSigma,NewComplexSigma),
   prove([[[X, A => B],[[X,C,Y] | Used]] | TempCond],
      [LitSigma,TransSigma,ComplexSigma],
      [LitDelta,[[X,A,Y]|TransDelta],ComplexDelta],Labels),
   prove([[[X, A => B],[[X,C,Y] | Used]] | TempCond],
      [NewLitSigma,TransSigma,NewComplexSigma],
      [LitDelta,TransDelta,ComplexDelta],Labels).
```

The predicate `put` is used to put `[Y,B]` in the proper sub-list of the antecedent. To search a derivation of a sequent $\Sigma \vdash \Delta$, CondLean proceeds as follows. First of all, if $\Sigma \vdash \Delta$ is an axiom, the goal will succeed immediately by using the clauses for the axioms. If it is not, then the first applicable rule will be chosen, e.g. if `ComplexDelta` contains a formula `[X,A -> B]`, then the clause for ($\rightarrow$ R) rule will be used, invoking `prove` on the unique premise of ($\rightarrow$ R). CondLean proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible.

CondLean extends the conditional language to formulas with connectives $\top$, $\wedge$, and $\vee$.

In systems containing the (CEM) rule, another parameter, called `CEM`, is added to the predicate `prove`. This parameter is needed in order to control the application of (CEM) by means of the restrictions stated by Lemmas 4.26 and 4.28, and it is used in the same way as `Cond` is used to control the applications of ($\Rightarrow$ L). More in detail, `CEM` is a list of pairs $[[X, A, Y], \mathit{Used}]$, where *Used* is a list of transitions $[[X, A_1, Y_1], ..., [X, A_n, Y_n]]$ such that the (CEM) rule has already been applied to $x \xrightarrow{A} y$ by using transitions $x \xrightarrow{A_i} y_i$. The clause for (CEM) is applied to a transition $x \xrightarrow{A} y$ by choosing $x \xrightarrow{A} z$ such that $x \xrightarrow{A_y} z$ belongs to `Sigma` and $[x, a_y, z]$ *does not belong to* `Used`.

As explained in Chapter 4, in order to describe a decision procedure for the presented conditional logics, we also need to control the application of (ID), (MP), and (CS) in systems allowing them; in particular, Lemmas 4.21, 4.22 and 4.34 state that it is useless to apply each of these rules more than once on the same transition $x \xrightarrow{A} y$ in the same branch of a proof tree. Consider the case of SeqID: in order to apply (ID) by means of this restriction, we add to the predicate `prove` a further argument, called `ID`, which is simply the *list of transitions* $x \xrightarrow{A} y$ to which the (ID) rule has already been applied in the current branch. The application of (ID) is therefore restricted to transitions *not belonging* to `ID`.
In systems comprising (MP) or (CS) we adopt the same strategy, by adding arguments `MP` or `CS`, respectively, to the predicate `prove`. In systems implementing combinations of presented axioms, the predicate `prove` is equipped with all the cor-

[1] There are other clauses implementing ($\Rightarrow$ L), taking care of cases when `Cond` is empty and when $x \xrightarrow{A} x$ is used (systems allowing MP only).

responding additional arguments; for instance, in the Prolog program implementing SeqCEM+ID+CS, the predicate `prove` has the following type:

```
prove(Cond,CEM,CS,ID,Sigma,Delta,Labels).
```

6.2.2 The Free Variables Version

For the basic system CK and its extension CK+ID, we have shown in Chapter 4, Lemma 4.47 and Theorem 4.19, that we can reformulate the crucial ($\Rightarrow$ L) rule as shown in Figure 4.7, i.e. with a non-invertible one, that is to say that the principal formula $x : A \Rightarrow B$ is not copied into the two premises of the rule. Moreover, considering a backward application of ($\Rightarrow$ L) to $\Gamma, x : A \Rightarrow B \vdash \Delta$, the left premise of the rule, i.e. the one in which a transition $x \xrightarrow{A} y$ is introduced in the right hand side of the sequent, can be restricted to a sequent of the form $x \xrightarrow{C} y \vdash x \xrightarrow{A} y$, such that $x \xrightarrow{C} y \in \Gamma$. This refinement is stated by the following Theorem:

Theorem 6.1. *SeqCK and SeqID are sound and complete even if ($\Rightarrow$ L) is formulated as follows:*

$$\frac{x \xrightarrow{C} y \vdash x \xrightarrow{A} y \qquad \Gamma, x \xrightarrow{C} y, y : B \vdash \Delta}{\Gamma, x \xrightarrow{C} y, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow L)$$

CondLean also implements this reformulated version of the calculi for CK{+ID}. The clause corresponding to the above non-invertible version of ($\Rightarrow$ L) is as follows:

```
prove([LitSigma,TransSigma,ComplexSigma],[LitDelta,TransDelta,
      ComplexDelta],Labels):-
   select([X,A => B],ComplexSigma,ResComplexSigma),
   member([X,C,Y],TransSigma),
   prove([[],[[X,C,Y]],[]],[[],[[X,A,Y]],[]],Labels),
   put([Y,B],LitSigma,ResComplexSigma,NewLitSigma,NewComplexSigma),
   prove([NewLitSigma,TransSigma,NewComplexSigma],[LitDelta,TransDelta,
      ComplexDelta],Labels).
```

The predicate `select` removes the formula `[X,A => B]` from the list `ComplexSigma`, then the conditional formula is not copied into the premises.

When the above ($\Rightarrow$ L) clause is used to prove $\Gamma, x : A \Rightarrow B \vdash \Delta$, a backtracking point is introduced by the choice of a label `Y` occurring in the two premises of the rule; in case of failure, Prolog's backtracking tries every instance of the rule with every available label (if more than one). Choosing, sooner or later, the right label to apply ($\Rightarrow$ L) may strongly affect the theorem prover's efficiency: if there are n labels to choose for an application of ($\Rightarrow$ L) the computation might succeed only after *n-1* backtracking steps, with a significant loss of efficiency.

Our second implementation, called *free variables*, makes use of *Prolog variables* to represent all the labels that can be used in a single application of the ($\Rightarrow$ L) rule. This version represents labels by integers starting from 1; by using integers we can easily express constraints on the range of the variable-labels. To this regard the library `clpfd` is used to manage free-variable domains. As an example, in order to prove $\Sigma^{'}$, *1:* $A \Rightarrow B \vdash \Delta$ the theorem prover will call `prove` on the following premises: $\Sigma^{'} \vdash \Delta$*, 1* $\xrightarrow{A}$ Y and *Y: B,* $\Sigma^{'} \vdash \Delta$, where Y is a Prolog variable. This variable will be then instantiated by Prolog's pattern matching to apply either the (EQ) rule, or to *close a branch with an axiom.* Here below is the clause implementing the ($\Rightarrow$ L) rule:

```
prove([LitSigma,TransSigma,ComplexSigma],[LitDelta,
      TransDelta,ComplexDelta],Max):-
   select([X,A => B],ComplexSigma,ResComplexSigma),
   domain([Y],1,Max), Y#>X,
   put([Y,B],LitSigma,ResComplexSigma,NewLitSigma,NewComplexSigma),
   prove([NewLitSigma,TransSigma,NewComplexSigma],
      [LitDelta,TransDelta,ComplexDelta],Max),
   prove([LitSigma,TransSigma,ResComplexSigma],
      [LitDelta,[[X,A,Y]|TransDelta],ComplexDelta],Max).
```

The atom `Y#>X` adds the constraint `Y>X` to the constraint store: the constraints solver will verify the consistency of it during the computation. In SeqCK and SeqID we can only use labels introduced *after* the label `X`, thus we introduce the previous constraint.

We have tested CondLean, CK system, on a valid sequent with 65 labels on the antecedent: the free variable version succeeds in less than 60 mseconds, whereas the constant labels version, even considering the reformulated one where ($\Rightarrow$ L) is reformulated as stated by Theorem 6.1, takes 460 mseconds.

The reformulation presented above is only applicable to systems SeqCK and SeqID, whereas in all the other systems we have to consider the invertible formulation of ($\Rightarrow$ L) given in Figure 4.1 and reported here for a better readability:

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow L)$$

Moreover, Lemmas 4.20, 4.27, and 4.33 allow to restrict the choice of the label y to use in the rule application in a way such that there exists $x \xrightarrow{C} y \in \Gamma$.
The clause implementing this version of ($\Rightarrow$ L) does not introduce a backtracking point, since the principal formula $x : A \Rightarrow B$ is copied in both the premises. However, if several transitions $x \xrightarrow{C_1} y_1, x \xrightarrow{C_2} y_2, \ldots, x \xrightarrow{C_n} y_n$ belong to Γ, then the choice of the label to use in the premises is crucial for CondLean's performances: indeed, choosing later the right label to use increases the dimension of the proof search space, affecting the performances of the theorem prover.

In order to avoid this problem, we have also developed a free variable version for *all* the systems of conditional logics presented here. The idea is the same as the free variable version described above for the systems SeqCK and SeqID: Prolog variables are used to represent all the labels that can be used in a single application of the ($\Rightarrow$ L) rule, and labels are represented by integers starting from 1. In order to prove $\Sigma^{'}$, *1:* $A \Rightarrow B \vdash \Delta$ the theorem prover will call `prove` on the following premises: $\Sigma^{'}, x : A \Rightarrow B \vdash \Delta, 1 \xrightarrow{A} Y$ and $\Sigma^{'}, x : A \Rightarrow B, Y : B \vdash \Delta$, where Y is a Prolog variable. This variable will be then instantiated by Prolog's pattern matching to apply either the (EQ) rule, or to close a branch with an axiom. In order to guarantee termination, we have to take care of Lemmas 4.18 and 4.20, that is to say:

- the domain of Y is restricted to the values of labels $y_1, y_2, \ldots, y_n$ such that

$x \xrightarrow{C_1} y_1, x \xrightarrow{C_2} y_2, \dots x \xrightarrow{C_n} y_n \in \Sigma'$;

- the predicate `prove` is equipped by the additional argument `Cond` described in the previous subsection, needed to avoid multiple applications of (⇒ L) with the same transition.

We present one of the clauses[2] implementing the invertible (⇒ L) in the free variable version:

```
prove(Cond,[LitSigma,TransSigma,ComplexSigma],
     [LitDelta,TransDelta,ComplexDelta],Max):-
  member([X,A => B],ComplexSigma),
  \+member([[X,A => B],_],Cond),
  y_domain(X,TransSigma,YDomain),
  list_to_fdset(YDomain,Y_FD_Domain),
  Y in_set Y_FD_Domain,
  put([Y,B],LitSigma,ComplexSigma,NewLitSigma,NewComplexSigma),
  prove([[[X,A => B],YDomain]|Cond],[NewLitSigma,TransSigma,NewComplexSigma],
     [LitDelta,TransDelta,ComplexDelta], Max),
  prove([[[X,A => B],YDomain]|Cond],[LitSigma,TransSigma,ComplexSigma],
     [LitDelta,[[X,A,Y]|TransDelta],ComplexDelta],Max).
```

The predicate `y_domain` returns a Prolog list, called `YDomain`, corresponding to the domain of `Y`, the free variable introduced by the rule application; for technical reasons, the predicate `list_to_fdset` is applied to convert this list into a set. The line `Y in_set Y_FD_Domain` updates the constraint store with the new free variable and its corresponding domain. In order to avoid further applications of (⇒ L) on `[X,A => B]` by using the same values for `Y`, a pair `[[[X,A => B],YDomain]` is added to the list `Cond`.

6.2.3 The Heuristic Version

We have also developed a third version, called *heuristic version*, that performs a "two-phase" computation:

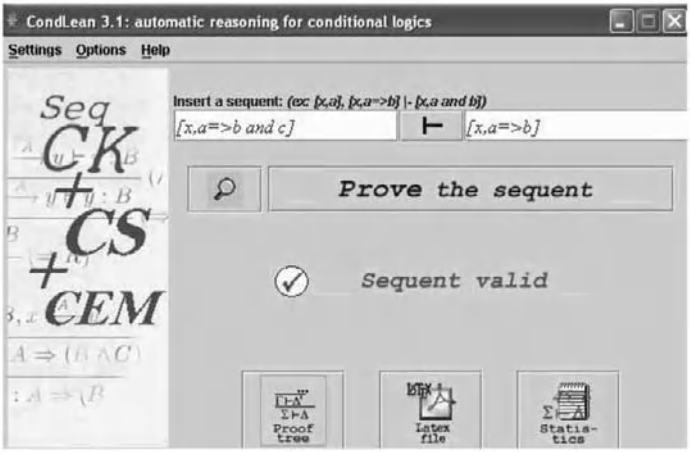
- in "Phase 1" an *incomplete* theorem prover searches a derivation exploring a *reduced search space*;
- in case of failure, the free-variables version is called ("Phase 2").

Intuitively, the reduction of the search space in Phase 1 is obtained by committing the choice of the label to instantiate a free variable, whereby blocking the backtracking.

6.2.4 CondLean's Graphical User Interface

The program CondLean has also a graphical interface (GUI) implemented in Java. The GUI interacts with the SICStus Prolog implementation by means of the package

[2] In order to increase readability, we do not present the other clause implementing (⇒ L), considering the case when a pair `[[X,A => B],`*Used*`]` belongs to `Cond`. CondLean's SICStus Prolog source code is available at `http://www.di.unito.it/∼pozzato/condlean 3.2/Prolog Source Code`.

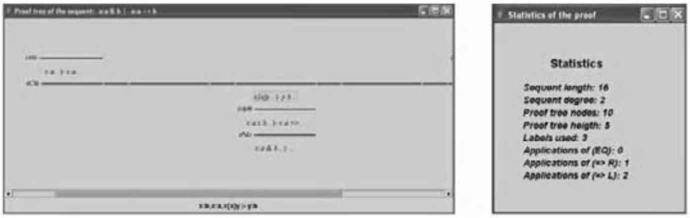


Insert a sequent: (ex [x,a], [x,a=>b] |- [x,a and b])

[x,a=>b and c] ⊢ [x,a=>b]

✓ Sequent valid

Figure 6.1: The control window of CondLean.



Statistics

Sequent length: 16
Sequent degree: 2
Proof tree nodes: 10
Proof tree heigth: 5
Labels used: 3
Applications of (EQ): 0
Applications of (=> R): 1
Applications of (=> L): 2

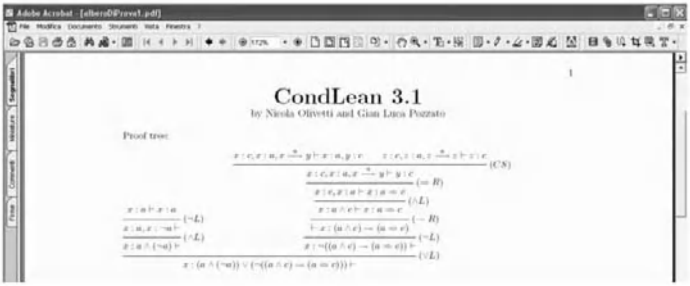
Figure 6.2: When the sequent is valid, the derivation found by the SICStus Prolog engine can be displayed in a special window. Users can also view some statistics of the proof.

`se.sics.jasper`. Thanks to the GUI, one does not need to know how to call the predicate `prove`: one just introduces a sequent in a text box and searches a derivation by clicking a button. Moreover, one can choose the intended system of conditional logic. Some pictures of CondLean are presented in Figures 6.1, 6.2, and 6.3.

The predicate `prove`, implementing SeqS calculi, is equipped by an additional argument, called `ProofTree`, which is used by the SICStus Prolog implementation in order to *return* the proof tree found back to the Java interface. When the user asks CondLean to prove a sequent by clicking the button, then the Prolog engine starts its work and, if the sequent is valid, it returns a functor `tree`, matching with the argument `ProofTree` and representing the derivation found. The functor `tree` is then manipulated by the Java interface in order to display the derivation in a special window.

When the submitted sequent is valid, CondLean offers these options:

- display a proof tree of the sequent in a special window;
- build a LaTeX source file containing the same proof tree: one can then compile it with a LaTeX engine and obtain a printable version of the tree (in a .dvi or .ps or .pdf file);
- view some statistics of the proof.


Adobe Acrobat - [alberoDiProva1.pdf]
File Modifica Documento Strumenti Vista Finestra ?
172%
Segnalibri
Miniature
Commenti
Firme
1

CondLean 3.1

by Nicola Olivetti and Gian Luca Pozzato

Proof tree:

$x : c, x : a, x \xrightarrow{a} y \vdash x : a, y : c$
$x : c, x : a, x \xrightarrow{a} x \vdash x : c$
(CS)
$x : c, x : a, x \xrightarrow{a} y \vdash y : c$
$(\Rightarrow R)$
$x : c, x : a \vdash x : a \Rightarrow c$
$(\wedge L)$
$x : a \wedge c \vdash x : a \Rightarrow c$
$(\rightarrow R)$
$\vdash x : (a \wedge c) \rightarrow (a \Rightarrow c)$
$(\neg L)$
$x : \neg((a \wedge c) \rightarrow (a \Rightarrow c)) \vdash$
$x : a \vdash x : a$
$(\neg L)$
$x : a, x : \neg a \vdash$
$(\wedge L)$
$x : a \wedge (\neg a) \vdash$
$(\vee L)$
$x : (a \wedge (\neg a)) \vee (\neg((a \wedge c) \rightarrow (a \Rightarrow c))) \vdash$

Figure 6.3: CondLean also builds a LaTeX source file containing the derivation found.

6.2.5 CondLean vs a "really-lean" Implementation

In the previous sections we have described CondLean, a theorem prover for some standard conditional logics. CondLean is *inspired* by the "lean" methodology, introduced by Beckert and Posegga [BP95] and analyzed by Fitting [Fit98]. In [BP95] an efficient and elegant first-order theorem prover, called leanT^AP, is presented. leanT^AP is a very short Prolog program consisting of only five clauses, and makes use of Prolog's search mechanism and backtracking as much as possible.

CondLean is only inspired to the "lean" methodology, but it does not fit its style in a rigorous manner. The main differences between our implementation and a "*really-lean*" one are the following:

- CondLean makes use of some auxiliary predicates, such as `put`, `select`, and `member`, whereas leanT^AP only relies on Prolog's clause indexing scheme and backtracking;
- the first argument of the predicate `prove` in leanT^AP is the next formula to be processed, which is always the leftmost formula in a single-sided sequent; this allows it to use the first-argument indexing refinements available in SICStus Prolog. CondLean does not present this characteristic, so it cannot take advantage of this refinement.

In this section we briefly present a possible "*really-lean*" implementation of SeqS calculi, in order to compare its performance with CondLean's. We have restricted our concern to the basic system CK, and considered a single-sided sequent calculus for it obtained from SeqCK. We have then implemented this single-sided sequent calculus in SICStus Prolog, obtaining a program called leanCK. leanCK is shorter than CondLean (without taking into account predicates used to generate new labels and to interact with the graphical interface, leanCK comprises only 19 clauses whereas CondLean needs 30 clauses); however, as we will discuss at the end of this section, leanCK's performance are worse than CondLean's.

We say that an atomic formula or its negation is a *literal*, i.e.formulas $x : P$, $x : \neg P$, with $P \in ATM$, are literals. We also represent with $\neg(x \xrightarrow{A} y)$ a transition formula $x \xrightarrow{A} y$ belonging to the right-hand side of a sequent.

We now consider sequents having the form $\Gamma \vdash \Lambda \mid\mid T$, where Γ is a multiset of labeled formulas, Λ is a multiset of literals, and T is a multiset of transition formulas. The basic idea is that complex formulas always belong to the left-hand side of the

$$(\mathbf{AX})\ \Gamma, x : P \vdash x : P, \Lambda \mid\mid T \qquad (\mathbf{AX})\ \Gamma, x : \neg P \vdash x : \neg P, \Lambda \mid\mid T$$

$$(\alpha\text{-rule})\ \frac{\alpha_1, \alpha_2, \Gamma \vdash \Lambda \mid\mid T}{\alpha, \Gamma \vdash \Lambda \mid\mid T} \qquad (\beta\text{-rule})\ \frac{(*)\beta_1, \Gamma \vdash \Lambda \mid\mid T \qquad (**)\beta_2, \Gamma \vdash \Lambda \mid\mid T}{\beta, \Gamma \vdash \Lambda \mid\mid T} \qquad \textbf{double negation}\ \frac{x : A, \Gamma \vdash \Lambda \mid\mid T}{x : \neg\neg A, \Gamma \vdash \Lambda \mid\mid T}$$

$$(\Rightarrow \mathbf{R})\ \frac{x \xrightarrow{A} y, y : \neg B, \Gamma \vdash \Lambda \mid\mid T}{x : \neg(A \Rightarrow B), \Gamma \vdash \Lambda \mid\mid T}\ (y \text{ new label}) \qquad (\Rightarrow \mathbf{L})\ \frac{(*)\neg(x \xrightarrow{A} y), x : A \Rightarrow B, \Gamma \vdash \Lambda \mid\mid T \qquad (**)\ y : B, x : A \Rightarrow B, \Gamma \vdash \Lambda \mid\mid T}{x : A \Rightarrow B, \Gamma \vdash \Lambda \mid\mid T}$$

$$(\mathbf{EQ})\ \frac{(*)u : A, u : \neg B \vdash \emptyset \mid\mid \emptyset \qquad (**)u : B, u : \neg A \vdash \emptyset \mid\mid \emptyset}{x \xrightarrow{A} y, \Gamma \vdash \Lambda \mid\mid x \xrightarrow{B} y, T} \qquad (\mathbf{EQ})\ \frac{(*)u : A, u : \neg B \vdash \emptyset \mid\mid \emptyset \qquad (**)u : B, u : \neg A \vdash \emptyset \mid\mid \emptyset}{\neg(x \xrightarrow{A} y), \Gamma \vdash \Lambda \mid\mid \neg(x \xrightarrow{B} y), T}$$

$$\textbf{Thinning for a literal}\ \frac{x : P \vdash \Lambda \mid\mid \emptyset}{x : P, \Gamma \vdash y : Q, \Lambda \mid\mid T} \qquad \textbf{Thinning for a literal}\ \frac{x : \neg P \vdash \Lambda \mid\mid \emptyset}{x : \neg P, \Gamma \vdash y : \neg Q, \Lambda \mid\mid T} \qquad \textbf{Thinning for a transition}\ \frac{x \xrightarrow{A} y \vdash \emptyset \mid\mid T}{x \xrightarrow{A} y, \Gamma \vdash \Lambda \mid\mid u \xrightarrow{B} w, T}$$

$$\textbf{Thinning for a transition}\ \frac{\neg(x \xrightarrow{A} y) \vdash \emptyset \mid\mid T}{\neg(x \xrightarrow{A} y), \Gamma \vdash \Lambda \mid\mid \neg(u \xrightarrow{B} w), T} \qquad \textbf{Duality for a literal}\ \frac{\Gamma \vdash x : \neg P, \Lambda \mid\mid T}{x : P, \Gamma \vdash \Lambda \mid\mid T} \qquad \textbf{Duality for a literal}\ \frac{\Gamma \vdash x : P, \Lambda \mid\mid T}{x : \neg P, \Gamma \vdash \Lambda \mid\mid T}$$

$$\textbf{Duality for a transition}\ \frac{\Gamma \vdash \Lambda \mid\mid x \xrightarrow{A} y, T}{x : \neg(x \xrightarrow{A} y), \Gamma \vdash \Lambda \mid\mid T} \qquad \textbf{Duality for a transition}\ \frac{\Gamma \vdash \Lambda \mid\mid \neg(x \xrightarrow{A} y), T}{x \xrightarrow{A} y, \Gamma \vdash \Lambda \mid\mid T}$$

Figure 6.4: Single-sided sequent calculus for CK.

sequent, whereas the right-hand side only contains literals and transitions, used to close a branch in the derivation and to apply the (EQ) rule, respectively.

For propositional formulas, we adopt the well-known α/β classification of Smullyan [Smu68], in which α's are conjunctions and β's are disjunctions:

α	α_1	α_2
$x : A \wedge B$	$x : A$	$x : B$
$x : \neg(A \vee B)$	$x : \neg A$	$x : \neg B$
$x : \neg(A \rightarrow B)$	$x : A$	$x : \neg B$

β	β_1	β_2
$x : \neg(A \wedge B)$	$x : \neg A$	$x : \neg B$
$x : A \vee B$	$x : A$	$x : B$
$x : A \rightarrow B$	$x : \neg A$	$x : B$

The single-sided sequent calculus for CK is presented in Figure 6.4. We consider $P, Q \in ATM$, and suppose that $P \neq Q$. Moreover, in the thinning for transitions, we suppose that $x \neq u$ or $y \neq w$. $(*)$ and $(**)$ are used to denote the two premises of branching rules.

We have then implemented the calculus in Figure 6.4 in SICStus Prolog. This program is called leanCK. As for CondLean, the calculus is implemented by a predicate `prove`, having the form

```
prove(Fml,UnExp,Lits,Trans,Labels,Cond).
```

where the first three arguments are defined as for propositional leanT^AP introduced by Fitting in [Fit98]. In particular, `Fml` is the formula currently being expanded in a given branch, `UnExp` is the list of formulas that have not yet been considered, and `Lits` is the list of literals occurring in that branch. `Trans` is the list of transitions (the multiset T in Figure 6.4). `Labels` is the list of labels occurring in that branch,

and Cond is a list of pairs [[X,A => B],*Used*] needed to control the application of (⇒ L) in the same manner as described for CondLean.
First, the program presents clauses for the classification of Smullyan:

```
type([X,A and B],conjunction,[X,A],[X,B]).
type([X,neg (A or B)],conjunction,[X,neg A],[X,neg B]).
...
type([X,A or B],disjunction,[X,A],[X,B]).
...
```

The rest of the very small Prolog code of leanCK consists in the clauses for the predicate prove. As an example, the clause implementing the $\beta-$rule is as follows:

```
prove(Fml,UnExp,Lits,Trans,Labels,Cond):-
   type(Fml,disjunction,Beta1,Beta2),!,
   prove(Beta1,UnExp,Lits,Trans,Labels,Cond),
   prove(Beta2,UnExp,Lits,Trans,Labels,Cond).
```

leanCK proceeds as follows: if Fml is a complex formula, then the corresponding clause is applied, and the proof search goes on with the recursive call(s) on the premise(s) of the rule. If Fml is a literal [X1,L1], leanCK first checks if the same literal [X1,L1] has already been added to the list Lits, in order to conclude the proof by the axioms; in detail:

1. if [X1,L1] corresponds to the head of Lits, then the computation succeeds;
2. otherwise, the thinning for a literal is applied, in order to find [X1,L1] in the tail of the list Lits;

If [X1,L1] does not belong to Lits, then its dual [X1,neg L1] is added to Lits, corresponding to an application of the rule of duality for a literal. Here are the clauses taking care of literals, implementing the machinery discussed above:

```
prove([X1,L1],_,[[X2,L2]|Lits],_,_,_):-
   (X1=X2,L1=L2,!)  ; prove([X1,L1],[],Lits,[],[],[]).
prove([X,L],[Next|UnExp],Lits,Trans,Labels,Cond):-!,
   prove(Next,UnExp,[[X,neg L]|Lits],Trans,Labels,Cond).
```

leanCK operates in a similar manner if Fml is a transition formula [X,A,Y] (resp. [neg,X,A,Y]); in particular, leanCK tries to find a transition [X,B,Y] (resp. [neg,X, B,Y]) belonging to the list Trans, in order to apply the (EQ) rule. In case of failure, leanCK moves the transition in Trans as [neg,X,A,Y] (resp. [X,A,Y]). The entire SICStus Prolog source code is available at

http://www.di.unito.it/∼pozzato/condlean3.2/
Prolog Source Code/leanck.pl

Let us briefly discuss about the implementation of the crucial rule (⇒ L). We have tried to respect the "lean" style as much as possible, however we have relaxed some constraints in order to implement the restrictions on the application of the rule (see Lemmas 4.18 and 4.20). Here below is the set of clauses used to implement (⇒ L):

```
prove([X,A => B],UnExp,Lits,Trans,Labels,Cond):-
   select([[X,A => B],Used],Cond,NewCond),!,
   (member([X,_,Y],UnExp) ; member([neg,X,_,Y],Trans)),
   \+member(Y,Used),
   append(UnExp,[[X,A => B]],NewUnExp),
   prove([Y,B],NewUnExp,Lits,Trans,Labels,
      [[[X,A => B],[Y|Used]]|NewCond]),
   prove([neg,X,A,Y],NewUnExp,Lits,Trans,Labels,
      [[[X,A => B],[Y|Used]]|NewCond]).
prove([X,A => B],UnExp,Lits,Trans,Labels,Cond):-
   (member([X,_,Y],UnExp) ; member([neg,X,_,Y],Trans)),
   append(UnExp,[[X,A => B]],NewUnExp),
   prove([Y,B],NewUnExp,Lits,Trans,Labels,
      [[[X,A => B],[Y]]|Cond]),
   prove([neg,X,A,Y],NewUnExp,Lits,Trans,Labels,
      [[[X,A => B],[Y]]|Cond]).
prove([X,A => B],[Next|UnExp],Lits,Trans,Labels,Cond):-!,
   append(UnExp,[[X,A => B]],NewUnExp),
   prove(Next,NewUnExp,Lits,Trans,Labels,Cond).
```

If ($\Rightarrow$ L) has already been applied to `[X,A => B]` in the current branch, then the first clause is invoked, and a label `Y` not belonging to `Used` (i.e. not yet used to apply the rule on that conditional formula) is selected; in order to ensure termination (Lemma 4.20 and Theorem 4.19), `Y` is such that there exists either a transition `[X,_,Y]` in `UnExp` or a transition `[neg,X,_,Y]`, representing $\neg(x \xrightarrow{C} y)$ for any C, in `Trans`. The principal formula `[X,A => B]` is then copied in the two premises of the rule *by appending it at the bottom of the* `UnExp` *list.*

The second clause is similar to the first one, and treats the case when ($\Rightarrow$ L) is applied to `[X,A => B]` for the first time in the current branch.

When the first argument `Fml` is a conditional formula $x : A \Rightarrow B$, leanCK tries to apply ($\Rightarrow$ L) by using a transition $x \xrightarrow{A} y$ such that $x \xrightarrow{C} y$ has been already introduced in the current branch. It could be the case that no transitions $x \xrightarrow{C} y$ are available; as an example, consider the case of `prove([x,a=>(b and c)],[[x,neg(a=>b)]],[],[],[x],[])`: ($\Rightarrow$ L) is not applicable, then we need to *postpone* its application after the application of ($\Rightarrow$ R) on `[x,neg(a=>b)]`.
To this aim, when the first two clauses above fail or are not applicable, then the third and last clause is invoked. It only appends the conditional formula being analyzed at the bottom of `UnExp`, and the computation goes on with the next formula to be expanded (if any).

To test the validity of a conditional formula F one queries leanCK with the following goal:

```
prove([x,neg F],[],[],[],[x],[]).
```

It can be shown that if a formula is valid, i.e. there is a proof of it in SeqCK, then leanCK ensures a terminating search; on the contrary, if a formula is not valid, then leanCK does not guarantee termination (with a negative answer) by the presence of the third clause for ($\Rightarrow$ L) discussed above. Consider the following example: one queries leanCK with the goal `prove([x,a=>a],[[x,a=>b]],[],[],[x],[])`; the third clause of ($\Rightarrow$ L) is applied, and the proof search goes on with the recursive call on `prove([x,a=>b],[[x,a=>a]],[],[],[x],[])`, but ($\Rightarrow$ L) is once again not applicable, then `prove([x,a=>a],[[x,a=>b]],[],[],[x],[])` is invoked, and so on, incurring in a loop. To avoid this situation, a loop-checking mechanism is needed.

However, we have not implemented any loop-checking machinery, which would obviously affects the performance of the theorem prover. At this point, we believe we have enough elements to compare leanCK with CondLean, and conclude that the *"really-lean"* solution is worse than the other.

We have tested both CondLean, constant labels version for CK, and leanCK (without loop checking) over 90 samples generated by modifying the samples from [BG97] and from [Vig00]; the following table reports, for each system, the number of proofs successfully completed in a fixed time limit:

	Time to succeed				
Implementation	$1ms$	$100ms$	$1s$	$2s$	$5s$
CondLean	80	81	83	84	86
leanCK	68	71	71	71	72

We have also tested CondLean over 100 sequents *not valid* in CK. CondLean concludes its proof search, answering that the sequent is not valid, in less than 2 seconds for 92 sequents over 100; for 90 of them, CondLean answers in less than 200 mseconds, whereas it requires only 100 mseconds for concluding proofs for 88 ones.

As a consequence, we believe that, in order to implement a labeled calculus like SeqS, a solution only *inspired* to the "lean" methodology as CondLean offers better performance than an implementation following this style in a rigorous manner. CondLean makes a systematic use of auxiliary predicates such as `member`, `select`, and `put` in order to control the derivation. Moreover, by partitioning formulas of each sequent into three sub-lists (atoms, transitions, complex formulas) and by using the `select` (resp. the `member`) predicate, we can have a better control of proof search. For instance we can postpone the application of branching rules (including the critical rule of left conditional) since we can choose the rule to apply by selecting the next complex formula to process, instead of processing always the leftmost one, as it happens in a *"really-lean"* implementation. This reduces the search space, increasing the theorem prover's performance.

6.3 GoalD$\mathcal{U}$CK: a Goal-Directed Theorem Prover for Conditional Logics

In this section we present GoalD$\mathcal{U}$CK, a very simple implementation of the calculi $\mathcal{U}$S' introduced in Chapter 4. GoalD$\mathcal{U}$CK is a SICStus Prolog program consisting of only eight clauses (resp. nine clauses in systems allowing MP), each one of them implementing a rule of the calculus[3], with the addiction of some auxiliary predicates and of a predicate implementing the Flat operation. The SICStus Prolog program only consists of 52 lines of code.

Here again, the goal-directed proof search is implemented by a predicate

[3] For technical reasons, GoalD$\mathcal{U}$CK split the rule ($\mathcal{U}$ **prop**) in two clauses, one taking care of applying it to the specific case of a goal $x : Q$ by using a clause $x : G \rightarrow Q$.

prove(Goal,Gamma,Trans,Labels).

which succeeds if and only if the goal G, represented by `Goal`, derives from the knowledge base Γ, represented by the list of world formulas `Gamma` and by the list of transition formulas `Trans`. Labeled formulas (both world and transition formulas) are represented by Prolog lists `[X,A]` and `[X,A,Y]`, respectively, exactly as in CondLean. `Labels` is the list of labels occurring in the current branch of the proof.

For instance, to prove if $x : A \Rightarrow B$ derives from the database $x : A \Rightarrow C \Rightarrow (\top \rightarrow B), x : B \Rightarrow ((A \Rightarrow B) \rightarrow C)$, one queries GoalD$\mathcal{U}$CK with the following goal:

```
prove([x,a => b],[[x, a => c => (true -> b)],
            [x,b => ((a => b) -> c)]],[],[x]).
```

As for CondLean, the predicate `prove` is also equipped with an additional argument `Tree`, instantiated by a functor `tree` and used to store the steps of a derivation and then give a justification of the proof. As we will discuss below, GoalD$\mathcal{U}$CK naturally implements a free variable version, by the presence of the clause implementing ($\mathcal{U}$ **prop**).

As mentioned above, each clause of the predicate `prove` implements an axiom or rule of the calculus $\mathcal{U}$S'. Here below we present the clauses implementing $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$, ($\mathcal{U}$ **prop**), and ($\mathcal{U}$ **trans**) as examples:

```
prove([X,A => G],Gamma,Trans,Labels):-
   generateLabel(Y,Labels),
   prove([Y,G],Gamma,[[X,A,Y]|Trans],[Y|Labels]).

prove([X,Q],Gamma,Trans,Labels):-
   member([Y,F=>(G->Q)],Gamma),
   atom(Q),
   prove([X,G],Gamma,Trans,Labels),
   extract(F,List),
   proveList(X,Y,List,Gamma,Trans,Labels).

prove([X,A,Y],_,Trans,_):-
   member([X,AP,Y],Trans),
   flat(A,FlattenedA),
   flat(AP,FlattenedAP),
   proveFlat([x,A],[],[],[x],x,FlattenedAP),
   proveFlat([x,AP],[],[],[x],x,FlattenedA).
```

The clause implementing $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ is very intuitive: if $A \Rightarrow G$ is the current goal, then the `generateLabel` predicate introduces a new label `Y`, then the predicate `prove` is called to prove the goal $y : G$ from a knowledge base enriched by the transition $x \xrightarrow{A} y$. Obviously, in the versions of GoalD$\mathcal{U}$CK supporting CK+ID{+MP}, i.e. implementing the goal-directed calculi $\mathcal{U}$ID{+MP}, this clause is replaced by the one implementing the rule $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$.

The clause implementing ($\mathcal{U}$ **prop**) proceeds as follows: first, it searches for a clause $y : F \Rightarrow (G \rightarrow Q)$ in the database Γ, then it checks if Q is an atom, i.e. if $Q \in ATM$; second, it makes a recursive call to `prove` in order to find a derivation for the goal $x : G$; finally, it invokes two auxiliary predicates, `extract` and `proveList`, having the following functions:

- `extract` builds a list of the form `[`A_1`,` A_2`, ...,` A_n`]`, where $F = A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n$;
- `proveList` invokes recursively the predicate `prove` in order to find a uniform proof for each goal $x_i \xrightarrow{A_{i+1}} x_{i+1}$ such that A_{i+1} belongs to the list generated by `extract`. The clauses implementing this predicate is presented here below:

```
proveList(K,Y,[A],Gamma,Trans,Labels,SubTree):-
   prove([Y,A,K],Gamma,Trans,Labels,SubTree).
proveList(X,Y,[A—Tail],Gamma,Trans,Labels,[SubTree—SubList]):-
   prove([K,A,X],Gamma,Trans,Labels,SubTree),
   proveList(K,Y,Tail,Gamma,Trans,Labels,SubList).
```

In order to prove a goal $x : Q$, if a clause $y : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$ belongs to the database, then the predicate `extract` builds the list `[`A_n`,...,`A_2`,`A_1`]`, and the predicate `proveList` is invoked. The first three arguments of this predicate represent the label of the current goal x, the label y of the clause of the database and the list built by `extract`, respectively. We can observe that the clause ($\mathcal{U}$ **prop**) is implemented using free variables for the labels. Thus, the implementation follows the "free-variables" style as the free variables version of CondLean. This is the most natural choice for a goal-directed procedure. Indeed, when `proveList` is applied, it first invokes `prove` in order to accomplish the goal $K_1 \xrightarrow{A_n} x$, where K_1 is a free variable, which will be instantiated by the Prolog's pattern matching to apply ($\mathcal{U}$ **trans**). Furthermore, `proveList` is recursively applied to invoke `prove` on a goal $K_2 \xrightarrow{A_{n-1}} K_1$, where K_2 is also a free variable, and so on, until a goal $y \xrightarrow{A_1} K_n$ is proved.

Given a goal `[X,A,Y]`, the clause implementing ($\mathcal{U}$ **trans**) is invoked. It first searches for a transition formula `[X,AP,Y]` in the database (i.e. in the list `Trans`), then it calls the predicate `flat` on both formulas `A` and `AP`; this predicate implements the Flat operation, building a *list of databases* obtained by flattening `A` (resp. `AP`) as defined in Definition 4.50.

In order to prove $\mathsf{Flat}(u : A) \vdash_{GD} u : AP$ and $\mathsf{Flat}(u : AP) \vdash_{GD} u : A$, another auxiliary predicate, called `proveFlat`, is finally invoked by the clause implementing ($\mathcal{U}$ **prop**); `proveFlat` recursively invokes the predicate `prove` by using *all different databases* built by `flat`.

The performance of GoalD$\mathcal{U}$CK are also promising. In the next section we present some experimental results, and we also compare GoalD$\mathcal{U}$CK's performance with CondLean's.

6.4 Statistics: CondLean and GoalD$\mathcal{U}$CK

The performances of CondLean are promising. We have tested it running SICStus Prolog 3.12.0 on an AMD Atholn XP 2400+ 512MB RAM machine, obtaining the following results: in less than 2 seconds, the constant labels version succeeds in 79 tests over 90, the free-variables one in 73 (but 67 in less than 10 mseconds), the heuristic version in 78 (70 in less than 500 mseconds). The test samples have been generated by modifying the samples from [BG97]. Considering the sequent-degree (defined as the maximum level of nesting of the $\Rightarrow$ operator) as a parameter, the free-variables version succeeds in less than 1 second for sequents of degree 11 and in less than 2 seconds for sequents of degree 15.

We have also tested CondLean, free-variables version for CK+MP, on a small machine[4], obtaining that it succeeds in less than 2 seconds for all sequents of degree 15 used in our tests, 700 ms for degree 10, and 5 ms for degree 2.

As described in Section 6.2.5, we have also made a comparison between CondLean and leanCK, a theorem prover for conditional logic CK following the "lean" style in a more rigorous manner. We have observed that CondLean offers better performances than leanCK's; intuitively, this can be explained by the fact that CondLean makes use of auxiliary predicates such as `member` and `select`, and partitions formulas in complex, atomic, and transition formulas: this allows it to have a better control on the evolution of a derivation, resulting in a more efficient theorem prover. A "*really-lean*" implementation does not seem to be the best choice for implementing labelled calculi like SeqS.

The table below shows the number of successes in searching a derivation of *valid* sequents with respect to a fixed time limit. We have tested all implemented versions for CK over other 90 test samples.

	Time to succeed				
Implementation	$1ms$	$100ms$	$1s$	$2s$	$5s$
CondLean - constant labels version	80	81	83	84	86
CondLean - const. labels - ($\Rightarrow$ L) not invertible	69	71	73	73	73
CondLean - free variable version	76	76	78	79	80
CondLean - heuristic version	78	78	79	80	80
leanCK	68	71	71	71	72

We have also tested CondLean over 100 sequents *not valid* in CK. CondLean concludes its proof search, answering that the sequent is not valid, in less than 2 seconds for 92 sequents over 100; for 90 of them, CondLean answers in less than 200 mseconds, whereas it requires only 100 mseconds for concluding proofs for 88 ones.

From the above table, it seems to be reasonable to conclude that the constant labels version offers better performance. This result could be a direct consequence of the structures of the tested sequents. However, we believe that the free variables version also deserves some interest; indeed, as described above in the paper, it offers better performance on sequents of high degree or presenting a high number of different labels in the antecedent (see for instance the example mentioned in Section 6.2.2).

The performances of GoalD$\mathcal{U}$CK are also promising. We have implemented a SICStus Prolog program testing both GoalD$\mathcal{U}$CK and CondLean, versions for CK, in order to compare their performances. This program randomly generates a database containing a specific set of formulas, obtained by combining a fixed set of propositional variables ATM. Moreover, the program builds a set of goals, whose cardinality is specified as a parameter, that can be either derivable or not from the generated database. Each goal is obtained from a set ATM° of variables which is a *subset* of the set ATM of variables in the database, in order to study situations in which several formulas in the database are *useless* to prove a goal.

Here below we present the experimental results we have obtained. Given a database, a set of goals to prove, and a fixed time limit, we present the following information for both GoalD$\mathcal{U}$CK and CondLean:

[4] We have obtained these results running SICStus Prolog 3.10.0 on a Pentium 166 MMX, 96 MB RAM machine.

- the number of executions ended with a success, that is to say: the goal is derivable from the database, and the theorem prover answers positively within the fixed time limit (column "positive answers");
- the number of executions ended with a finite failure, that is to say: the goal is *not* derivable from the database, and the theorem prover answers negatively within the fixed time limit (column "negative answers");
- the number of executions ended with a time out, that is to say the theorem prover is not able to conclude its work within the time limit (column "timeouts").

Test A

- 100 goals
- Number of propositional variables ATM in the database: 9
- Number of propositional variables $ATM^\circ \subseteq ATM$ in the goals: 3
- Database size: 100 formulas
- Time limit: 1 ms

Theorem prover	positive answers	negative answers	timeouts
CondLean	72	0	28
GoalD$\mathcal{U}$CK	81	16	3

Test B

- 100 goals
- Number of propositional variables ATM in the database: 9
- Number of propositional variables $ATM^\circ \subseteq ATM$ in the goals: 2
- Database size: 200 formulas
- Time limit: 100 ms

Theorem prover	positive answers	negative answers	timeouts
CondLean	68	4	28
GoalD$\mathcal{U}$CK	65	11	24

Test C

- 100 goals
- Number of propositional variables ATM in the database: 20
- Number of propositional variables $ATM^\circ \subseteq ATM$ in the goals: 3
- Database size: 200 formulas
- Time limit: 100 ms

Theorem prover	positive answers	negative answers	timeouts
CondLean	47	0	53
GoalD$\mathcal{U}$CK	67	16	17

As expected, the above tests suggest that GoalD$\mathcal{U}$CK offers better performances when the database contains several clauses which are useless to derive a goal (as mentioned, this situation is implemented by imposing that the set of variables occurring in the goals is a subset of the set of variables in the database). In Test A, GoalD$\mathcal{U}$CK is able to prove 81 goals, whereas CondLean answers positively in 72 cases. Moreover, GoalD$\mathcal{U}$CK concludes its work in 97 cases over 100 within the fixed time limit of 1 ms, even with a finite failure in 16 cases, whereas CondLean results in a time out in 28 cases.

In Test B, CondLean seems to offer better performances in discovering derivable goals: it succeeds in 68 cases, against 65 of GoalD$\mathcal{U}$CK. Surprisingly enough, in this case GoalD$\mathcal{U}$CK runs faster in answering negatively, finding 11 not derivable goals, against 4 by CondLean.

In Test C, we have tested the theorem provers with a database containing 200 formulas and 20 propositional variables, whereas goals are built from a subset of ATM of cardinality 3. In this case, GoalD$\mathcal{U}$CK offers better performances, proving 20 goals more than CondLean in 100 ms.

We conclude this section by remarking that goal-directed proof methods usually do not ensure a terminating proof search. GoalD$\mathcal{U}$CK does not ensure termination too. Indeed, given a database Γ and a goal $x : G$, it could happen that, after several steps, the goal-driven computation leads to search a derivation for the same goal $x : G$ from a database Γ' such that $\Gamma' \supseteq \Gamma$ (that is to say: Γ' either corresponds to Γ or it is obtained from Γ by adding new facts). This problem is also well known in standard logic programming. As an example, consider the database containing the following fact:

$$x : (Q \wedge \top) \rightarrow Q$$

Querying GoalD$\mathcal{U}$CK with the goal $x : Q$, one can observe that the computation does not terminate: GoalD$\mathcal{U}$CK tries to apply ($\mathcal{U}$ **prop**) by using the only clause of the program (database), then it searches a derivation of the two subgoals $x : \top$ (and the computation succeeds) and $x : Q$. In order to prove this goal, GoalD$\mathcal{U}$CK repeats the above steps, then incurring in a loop.

This justifies the fact that, in the statistics above, GoalD$\mathcal{U}$CK produces several time outs, especially when the database contains a large number of clauses, increasing the probability of having a loop.

6.5 KLMLean: a Theorem Prover for KLM Logics

In this section we present an implementation of the tableaux calculi for KLM logics introduced in Chapter 5. Obviously, our theorem prover implements the terminating calculi for KLM systems, that here we denote with $\mathcal{TK}^{\mathbf{T}}$, where $K \in \{\mathbf{R}, \mathbf{P}, \mathbf{CL}, \mathbf{C}\}$.

The theorem prover, called **KLMLean**, is a SICStus Prolog program following the same guidelines of CondLean, that is to say:

- it is inspired to the "lean" methodology, even if it does not fit this style in a rigorous manner;

- it consists of a set of clauses, each one of them representing a tableau rule or axiom;
- the proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism;
- it comprises a graphical interface written in Java.

We represent each node of a proof tree by a *list* of formulas. Exactly as in CondLean, the tableaux calculi are implemented by the predicate

prove(Gamma,Sigma,Tree).

which succeeds if and only if Γ is unsatisfiable, where `Gamma` is the list representing the set of formulas Γ. `Sigma` is the list representing the set Σ of *used conditionals*, and it is used in order to control the application of the $(\mid\!\sim^{+})$ rule only to unused conditional, as described in Chapter 5.

If the predicate succeeds, `Tree` matches with a functor `tree` representing the closed tableau found by the reasoning system; this functor is used by the Java interface to display the proof.

Let us first describe the implementations of $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ and $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$, which are *non-labelled* calculi. Then we will describe an implementation of $\mathcal{T}\mathbf{C}$, ensuring termination by performing a simple loop-check machinery. We will conclude this section by introducing the version of KLMLean implementing the labelled calculus $\mathcal{T}\mathbf{R}^{\mathbf{T}}$.

6.5.1 KLMLean for P and CL

In Chapter 5 we have introduced non-labelled and terminating calculi $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ and $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$. The implementation of these calculi is very simple, since, as a difference with CondLean, we do not have to represent labelled formulas and "links" between labels (e.g. the transition formulas manipulated by CondLean).

For instance, to prove that $A \mid\!\sim B \wedge C, \neg(A \mid\!\sim C)$ is unsatisfiable in $\mathbf{P}$, one queries KLMLean with the following goal:

```
prove([a => (b and c), neg (a => c)],[ ],Res)).
```

The string `=>` is used to represent the conditional operator $\mid\!\sim$, `and` is used to denote $\wedge$, and so on. Each clause of `prove` implements one axiom or rule of the tableaux calculi; for example, the clause implementing $(\mid\!\sim^{+})$ is as follows:

```
prove(Gamma,Sigma,tree(entP,A,B,ST1,ST2)):-
   select(A => B,Gamma,NewGamma),!,
   prove([neg box neg A|NewGamma],[A => B|Sigma],ST1),!,
   prove([A -> B|NewGamma],[A => B|Sigma],ST2).
```

In the clause reported above, it is easy to understand how KLMLean implements the controlled application of $(\mid\!\sim^{+})$ on a conditional formula $A \mid\!\sim B$ (Theorem 5.14): $A \mid\!\sim B$ is moved from Γ to the set Σ of used conditionals, therefore the above clause will not further be applied to it (it only applies to the other conditionals belonging to Γ). Notice that KLMLean implements the following version of the $(\mid\!\sim^{+})$ rule, comprising only two conclusions:

$$\frac{\Gamma, A \mid\!\sim B; \Sigma}{\Gamma, A \rightarrow B; \Sigma, A \mid\!\sim B \qquad \Gamma, \neg\Box\neg A; \Sigma, A \mid\!\sim B}\ (\mid\!\sim^{+})$$

Obviously, the above one is equivalent to the ($\mid\!\sim^{+}$) rule, comprising three conclusions, presented in Chapter 5. KLMLean adopts this alternative version of the rule in order to simplify the Java interface (only binary trees are displayed in windows showing a closed tableau).

To search a closed tree of a set Γ of formulas of $\mathcal{L}$, KLMLean proceeds as follows. First of all, if Γ is an axiom, i.e. there is a formula F such that both `F` and `neg F` belong to `Gamma`, the goal will succeed immediately by using the clauses for the axioms. If it is not, then the first applicable rule will be chosen, e.g. if `Gamma` contains a formula `[A and B]`, then the clause for the ($\wedge^{+}$) rule will be used, invoking `prove` on the unique conclusion of ($\wedge^{+}$). KLMLean proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the branching rules and of the conditional and box rules is postponed as much as possible, according to the restriction in Definition 5.13; in detail, the clause implementing ($\mid\!\sim^{+}$) is the only one to be postponed to the one implementing ($\Box^{-}$).

6.5.2 KLMLean for $\mathbf{C}$

In Chapter 5, Section 5.4, we have introduced $\mathcal{T}\mathbf{C}$, a tableau calculus for the weakest logic of the KLM framework, namely Cumulative logic $\mathbf{C}$. By Theorem 5.40, we can describe a complete theorem prover without implementing the (Weak-Cut) rule. However, this is not enough to ensure termination; we also need to provide a loop checking procedure in order to avoid that a given set of formulas is expanded more than once on a branch, thus producing infinite branches.

We adopt a very simple solution: intuitively, the `prove` predicate is equipped with an additional argument, called `Analyzed`, used to represent the set of nodes already considered in the current branch of the derivation.
In order to obtain an efficient mechanism, KLMLean adopts services of the library `ordsets`. Each node of a tableau is represented by an *ordered set*, which is a list whose elements are ordered in a standard order. The ordering is defined by a SICStus Prolog family of term comparison predicates and it is the ordering produced by the built-in predicate `sort/2` (see [otSIoCS06] for details about SICStus Prolog's order sets). The set of nodes already analyzed in the current branch is also represented by an ordered set.

To search a derivation for a set of formulas Γ, having already considered the nodes belonging to the ordered set `Analyzed`, KLMLean proceeds as follows:

- first, it checks if the order set corresponding to Γ belongs to `Analyzed`;
- if Γ has not already been considered (i.e. it does not belong to `Analyzed`), then the "standard" proof search procedure is executed, i.e. if Γ is an axiom, the goal will succeed immediately by using the clauses for the axioms, if it is not, then the first applicable rule will be chosen, and so on; if Γ belongs to `Analyzed`, a loop has been detected, and the computation goes on considering unexplored alternatives (if any) since Prolog's backtracking is performed;
- when a clause implementing a rule of $\mathcal{T}\mathbf{C}$ is selected, the proof search procedure is recursively called on the conclusion(s) by adding Γ to `Analyzed`.

The following clause of the predicate `prove` implements the loop-checking machinery described above:

```
prove(Gamma,Sigma,Tree,Analyzed):-
    append(Gamma,Sigma,Node),
    list_to_ord_set(Node,Current),
    \+ord_member(Current,Analyzed),
    ord_add_element(Analyzed,Current,NewAnalyzed),
    proveStandard(Gamma,Sigma,Tree,NewAnalyzed).
```

The predicate `list_to_ord_set` is used to convert the current node of the tableau (the list obtained as the result of appending the list of used conditionals Σ to the list representing Γ) in the corresponding ordered set, called `Current`. The predicate `ord_member` checks if `Current` belongs to the order set `Analyzed` of nodes already considered in the current branch. If the node is analyzed for the first time, then the predicate `proveStandard` is invoked; this predicate implements the "standard" proof search procedure, i.e. each clause of it implements a tableau rule or axiom of $\mathcal{T}\mathbf{C}$. When the predicate `proveStandard` is called, then the set of nodes already considered in the current branch is updated with the current node `Current` by calling the predicate `ord_add_element`.

As an example, here below we present the clause of the predicate `proveStandard` implementing the rule ($\mid\!\sim^+$), "embedding" the cut on the antecedents of conditional formulas:

```
proveStandard(Gamma,Sigma,tree(...,ST1,ST2),Analyzed):-
    select(A => B,Gamma,NewGamma),!,
    conditionals(NewGamma,Cond),
    inboxed(NewGamma,InBoxed),
    append(Cond,InBoxed,Gammastar),
    append(Gammastar,Sigma,DefGammaLeft),
    prove([box neg l A|[l A|[A => B|DefGammaLeft]]],[],
          ST1,Analyzed),
    prove([(l A) -> (l A and box neg l A and l B)|NewGamma],
          [A => B|Sigma],ST2,Analyzed).
```

If `Gamma` contains a formula `[A => B]`, then the above clause is used. The predicate `inboxed` is used to compute the multiset $\Gamma^{\Box^{\downarrow}}$. The predicate `prove` (notice: not `proveStandard`) is invoked on the conclusions of the ($\mid\!\sim^+$) rule; in this way, the loop-checking mechanism described above will be applied at each step, before choosing the next clause to apply.

Notice that, also in this case, we have adopted an alternative, equivalent version of the ($\mid\!\sim^+$) rule, comprising only two conclusions. The rule adopted by KLMLean is as follows:

$$\frac{\Gamma, A \mid\!\sim B; \Sigma}{\Sigma, \Gamma^{\mid\sim^{\pm}}, \Gamma^{\Box^{\downarrow}}, A \mid\!\sim B, LA, \Box\neg LA; \emptyset \qquad \Gamma, LA \rightarrow (LA \wedge \Box\neg LA \wedge LB); \Sigma, A \mid\!\sim B}\ (\mid\!\sim^+)$$

As in the other cases, the choice of using an alternative version of the ($\mid\!\sim^+$) rule is only needed to simplify the implementation of the Java graphical interface.

6.5.3 KLMLean for $\mathbf{R}$

In this section we describe the implementation of the *labelled* tableau calculus $\mathcal{T}\mathbf{R}^{\mathbf{T}}$. The SICStus Prolog program is very similar to the one proposed for $\mathbf{P}$, provided the necessary adaptation to manipulate labelled formulas.

As CondLean, KLMLean makes use of Prolog constants to represent the labels of the calculus. $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ distinguishes two kinds of labelled formulas, namely *world* and *relation* formulas. They are represented by KLMLean as follows:

- a world formula $x : A$ is represented by a pair `[x,a]`;
- a relation formula $x < y$ is represented by a list `[y,<,x]`[5].

Exactly as for $\mathbf{P}$ (and for $\mathbf{CL}$), each clause of the predicate `prove` implements a tableau rule or axiom. As an example, here below is the clause implementing the rule $(<)$, capturing the modularity of the preference relation:

```
prove(Gamma,Labels,Cond,tree(...,LeftTree,RightTree)):-
    member([X,<,Y],Gamma),
    member(Z,Labels),
    X\=Z,
    Y\=Z,
    \+member([X,<,Z],Gamma),
    \+member([Z,<,Y],Gamma),!,
    gammaM(Gamma,Y,Z,ResLeft),
    gammaM(Gamma,Z,X,ResRight),
    append(ResLeft,Gamma,LeftConcl),
    append(ResRight,Gamma,RightConcl),
    prove([[Z,<,Y]|LeftConcl],Labels,Cond,LeftTree),!,
    prove([[X,<,Z]|RightConcl],Labels,Cond,RightTree).
```

If a relation formula `[x,<,y]` belongs to `Gamma`, then the above clause is selected. The predicate `member` nondeterministically chooses a label `Z` occurring in the current branch (i.e., belonging to the list `Labels`), then, if the side condition of the rule is satisfied (neither `[X,<,Z]` nor `[Z,<,Y]` belong to `Gamma`), the predicate `prove` is invoked on the two conclusions. The predicate `gammaM` is used to compute the multisets $\Gamma^{M}_{y \rightarrow z}$ and $\Gamma^{M}_{z \rightarrow x}$.

In this system, the list `Cond`, used to control the application of the $(\mid\!\sim^{+})$ rule, is a list of pairs of the form `[`*Label*`,`*Conditional formula*`]`. If $(\mid\!\sim^{+})$ has been applied to $A \mid\!\sim B$ in the current branch by using the label x, then `[x,a => b]` belongs to `Cond`.

As another difference with the implementations for the other KLM systems, in order to increase its performances, KLMLean for $\mathbf{R}$ adopts an heuristic approach (not "lean") to implement the crucial $(\mid\!\sim^{+})$ rule. Here below is the clause implementing $(\mid\!\sim^{+})$:

```
prove(Gamma,Labels,Cond,tree(...,LeftTree,RightTree)):-
   maxDegree(Gamma,Labels,Cond,[U,A => B],Deg),
    Deg > 0,
```

[5] We have decided to represent $x < y$ with a triple, rather than with a pair `[x,y]`, because this alternative solution would cause some implementation mistakes. Indeed, a relation formula $x < y$ should be confused with a world formula $x : y$.

```
prove([[U,A -> B]|Gamma],Labels,
     [[U,A => B]|Cond],LeftTree),!,
prove([[U,neg box neg A]|Gamma],Labels,
     [[U,A => B]|Cond],RightTree).
```

The predicate `prove` *chooses the "best" positive conditional* to which apply the rule, rather than selecting the leftmost one by using the `member` (or `select`) predicate. To this purpose, the predicate `maxDegree` is invoked. `maxDegree` proceeds as follows:

1. it creates a list, called `ListOfConditionals`, containing all possible combinations among labels and positive conditionals occurring in the branch; for instance, if x, y, and z are the labels, and $A \mid\sim B$ and $C \mid\sim D$ are the positive conditionals, then it returns the list `[[x,a=>b],[y,a=>b],[z,a=>b],[x,c=>d],[y,c=>d],[z,c=>d]]`;

2. it removes from the above `ListOfConditionals` all the items belonging to `Cond`, i.e. if $A \mid\sim B$ has already been expanded in the current branch by using x, then `[x,a=>b]` will be removed; we call `ListOfAvailableConditionals` the list resulting from this operation;

3. for each element of `ListOfAvailableConditionals`, it evaluates a *degree of usefulness*, then selects the combination label-conditional having the highest degree.

The predicate `prove` is finally recursively invoked on the two conclusions of $(\mid\sim^+)$, corresponding to an application of the rule on the "best" positive conditional, by using the "best" label, both selected by the `maxDegree` predicate.

Let Γ be the set of formulas currently analyzed by the predicate `prove`. The degree of usefulness of $x : A \mid\sim B$, intended as the degree of usefulness of an application of $(\mid\sim^+)$ on $A \mid\sim B$ by using the label x and denoted as $du(x : A \mid\sim B)$, is defined (and computed) as follows:

- if either $x : \neg A \in \Gamma$ or $x : \neg\Box\neg A \in \Gamma$ or $x : B \in \Gamma$, then the application of $(\mid\sim^+)$ is useless, since at least one of the conclusions will be identical to the premise; therefore, in this case we have that $du(x : A \mid\sim B) = 0$;
- otherwise, $du(x : A \mid\sim B) = du_1(x : A \mid\sim B) + du_2(x : A \mid\sim B) + du_3(x : A \mid\sim B)$, where:

$$du_1(x : A \mid\sim B) = \begin{cases} 12 \text{ if } x : \Box\neg A \in \Gamma \\ 1 \text{ otherwise} \end{cases}$$

$$du_2(x : A \mid\sim B) = \begin{cases} 5 \text{ if } x : A \in \Gamma \\ 1 \text{ otherwise} \end{cases}$$

$$du_3(x : A \mid\sim B) = \begin{cases} 5 \text{ if } x : \neg B \in \Gamma \\ 1 \text{ otherwise} \end{cases}$$

Intuitively, $du(x : A \mid\sim B)$ represents the "power of closure" of an application of $(\mid\sim^+)$, in order to find a derivation for Γ. If $x : A \in \Gamma$, $x : \Box\neg A \in \Gamma$ and $y : B \in \Gamma$, then obviously an application of $(\mid\sim^+)$ on $A \mid\sim B$ by using x is more powerful than by using y, in the sense that two conclusions (those containing $x : \neg A$ and $x : \neg\Box\neg A$, respectively) will lead to a closed tableau. We believe that an application of $(\mid\!\sim^+)$ closing the branch where $\neg\Box\neg A$ is introduced can be considered more promising than another one closing the other two branches: this justifies our arbitrary choice of assigning 12 to the maximum value of du_1, whereas the maximum value of du_2 and du_3 is set to 5.

KLMLean for **R** offers a further functionality, not implemented for the other systems. If the initial set of formulas Γ is *satisfiable*, i.e. the predicate `prove` fails, then the theorem prover builds a rational model satisfying Γ, in order to justify its negative response about unsatisfiability[6]. More in detail, if the predicate `prove` fails, then another predicate, called `proveCounter`, is invoked. Each clause of this predicate also implements axioms and rules of $\mathcal{T}\mathbf{R}^{\mathbf{T}}$; moreover, the following further clause is added:

```
proveCounter(Gamma,_,Labels):-
    listOfAtoms(Gamma,Atoms),
    listOfTransitions(Gamma,Relations),
    ⋮
    throw(saturatedBranch(Atoms,Labels,Relations,Ext)).
```

This additional clause is invoked when no rule is applicable to the current set of formulas, i.e. when the theorem prover has generated an open, saturated branch. `Atoms`, `Labels` and `Relations` are the lists representing the atomic formulas, the labels, and the relation formulas, respectively, occurring in that branch; these information are used to build a rational model satisfying the initial set of formulas. The predicate `proveCounter` also throws an exception called `saturatedBranch`, which is captured by the predicate used to link the SICStus Prolog engine with the Java graphical interface. The Java interface is therefore able to give a graphical representation of the rational model built by the theorem prover.

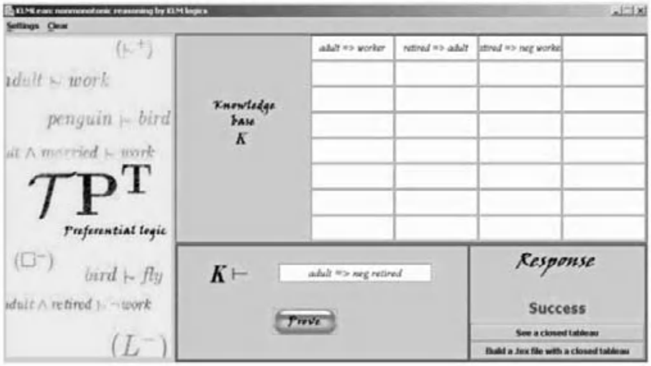
6.5.4 KLMLean's Graphical User Interface

As mentioned, KLMLean has also a graphical user interface (GUI) implemented in Java. As for CondLean, the GUI interacts with the SICStus Prolog implementation by means of the package `se.sics.jasper`. Thanks to the GUI, one does not need to know how to call the predicate `prove`, or if the program implements a labelled or an unlabelled deductive system; therefore, one just introduces

- the formulas of a knowledge base K (or the set of formulas to prove to be unsatisfiable);
- a formula F, in order to prove if one can infer F from K, corresponding to verify if $K \cup \{\neg F\}$ is unsatisfiable

in a text box and searches a derivation by clicking a button. Moreover, one can choose the intended system of KLM logic, namely **R**, **P**, **CL** or **C**. When the analyzed set $K \cup \{\neg F\}$ of formulas is unsatisfiable, KLMLean offers these options:

[6] To build a model for a satisfiable set of formulas is quite easy by considering a labelled calculus. This is the reason why we have implemented this function only in **R**.

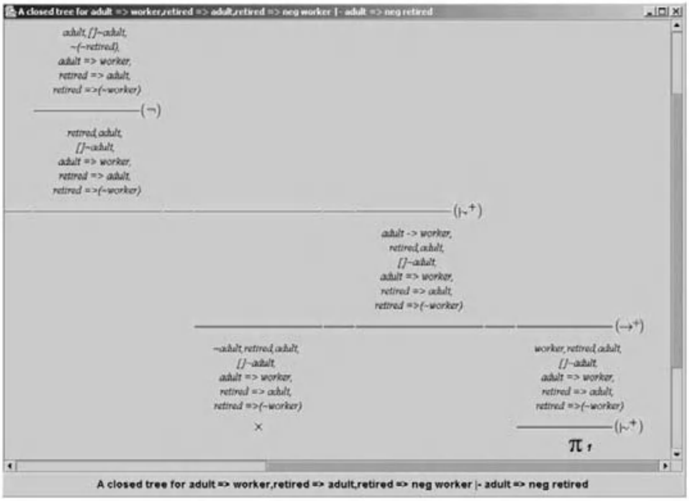

KLMLean: nonmonotonic reasoning by KLM logics

Knowledge base K: adult => worker | retired => adult | tired => neg worke

K ⊢ adult => neg retired

Response: Success

Figure 6.5: The main window of KLMLean


A closed tree for adult => worker,retired => adult,retired => neg worker |- adult => neg retired

A closed tree for adult => worker,retired => adult,retired => neg worker |- adult => neg retired

Figure 6.6: A proof tree in a special window.

- display a proof tree of the set in a special window;
- build a latex file containing the same proof tree: compiling this file with Latex, one can obtain the closed tree in a pdf file, or ps, or dvi, and then print it.

Some pictures of KLMLean are presented in Figures 6.5, 6.6, and 6.7.

As mentioned in Section 6.5.3, when the user executes KLMLean - implementation of $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ - on a satisfiable set of formulas Γ (i.e. the computation results in a failure), a graphical representation of a rational model satisfying Γ can be displayed in a special window. This representation consists in a graph, whose nodes represent possible worlds in the model and edges represent the preference relation among them. As an example, if one queries KLMLean, version for rational logic $\mathbf{R}$, in order to check whether $adult \wedge retired \mid\sim worker$ is entailed by $adult \mid\sim worker, retired \mid\sim adult, retired \mid\sim \neg worker, \neg(adult \mid\sim \neg married)$, then KLMLean shows the counterexample in Figure 6.8, since the set of formulas $adult \mid\sim worker, retired \mid\sim adult$,

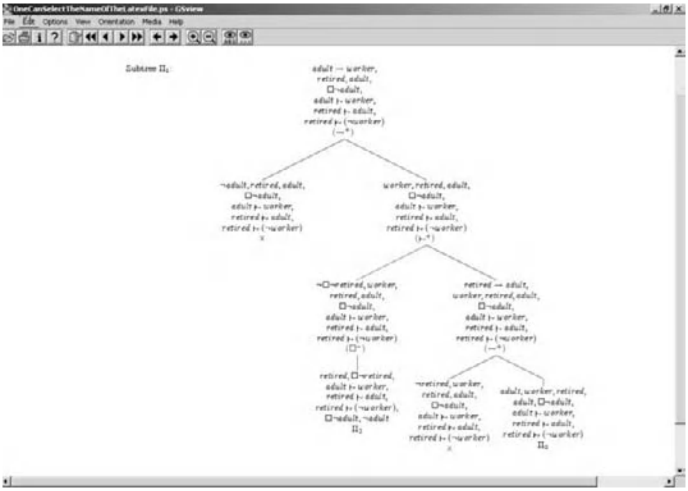

OneCanSelectTheNameOfTheLatexFile.ps - GSview
File Edit Options View Orientation Media Help
Subtree Π₁:
adult → worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
(→⁺)
¬adult, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
x
worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
(|∼⁺)
¬□¬retired, worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
(□⁻)
retired → adult, worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
(→⁺)
retired, □¬retired, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker), □¬adult, ¬adult
Π₂
¬retired, worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
x
adult, worker, retired, adult, □¬adult, adult |∼ worker, retired |∼ adult, retired |∼ (¬worker)
Π₃

Figure 6.7: The proof tree printed in a .ps file.

$retired \mid\!\sim \neg worker$, $\neg(adult \mid\!\sim \neg married)$, $\neg(adult \wedge retired \mid\!\sim worker)$ is satisfiable.

6.5.5 Performances of KLMLean

We have implemented a SICStus Prolog program generating random sets of sample formulas, then we have tested KLMLean over the generated sets.

We have obtained the following experimental results running SICStus Prolog 3.12.0 on a Mobile Intel Pentium 4, 2.00GHz, 512MB RAM machine. Each element of a table contains triples of the form

$$\langle n_s, n_f, n_t \rangle$$

where:

- n_s is the number of computations ending with a *success*, i.e. KLMLean answers that the initial set of formulas is *unsatisfiable* within the fixed time limit;
- n_f is the number of computations ending with a *failure*, i.e. KLMLean answers that the initial set of formulas is *satisfiable* within the fixed time limit;
- n_t is the number of *timeouts*, i.e. KLMLean is not able to give an answer within the fixed time limit.

Columns are labelled by the time limits adopted in the proofs. We present the experimental results in detail.

Test of KLMLean, Cumulative logic C

We have tested the implementation of the calculus $\mathcal{T}\mathbf{C}$ over 300 samples. Rows in the table below are labelled with pairs (Vars - Number of formulas), representing

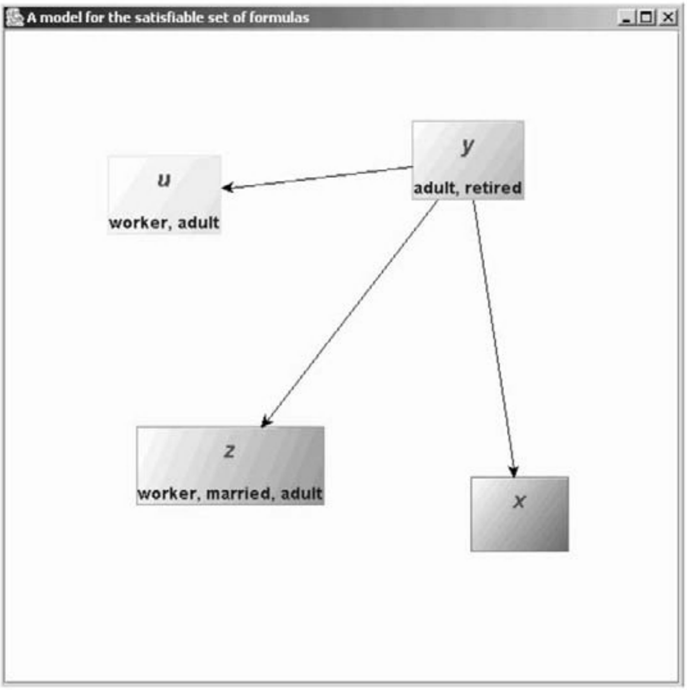


Figure 6.8: The model built by KLMLean - rational logic **R** - for the satisfiable set of formulas $adult \mid\!\sim worker, retired \mid\!\sim adult, retired \mid\!\sim \neg worker, \neg(adult \mid\!\sim \neg married), \neg(adult \wedge retired \mid\!\sim worker)$.

the number of different propositional variables and the number of formulas (boolean combinations of conditionals) occurring in the initial set, respectively. In detail, for each choice of number of propositional variables - number of formulas, we have tested KLMLean over 100 examples, obtaining the following results:

Vars - Num of fml	1 ms	10 ms	100 ms	1 s	2.5 s
3-5	$\langle 19, 28, 53\rangle$	$\langle 19, 29, 52\rangle$	$\langle 20, 38, 42\rangle$	$\langle 20, 51, 29\rangle$	$\langle 20, 55, 25\rangle$
5-10	$\langle 11, 0, 89\rangle$	$\langle 11, 0, 89\rangle$	$\langle 14, 2, 84\rangle$	$\langle 16, 3, 81\rangle$	$\langle 16, 9, 75\rangle$
7-25	$\langle 18, 0, 82\rangle$	$\langle 18, 0, 82\rangle$	$\langle 18, 0, 82\rangle$	$\langle 20, 0, 80\rangle$	$\langle 21, 2, 77\rangle$

Test of KLMLean, Loop-Cumulative logic CL

As for **C**, we have tested the implementation of the calculus $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ over 300 different sets of boolean combinations of conditionals, obtaining the following results:

Vars - Num of fml	1 ms	10 ms	100 ms	1 s	2.5 s
3-5	$\langle 48, 0, 52\rangle$	$\langle 48, 0, 52\rangle$	$\langle 59, 1, 40\rangle$	$\langle 62, 1, 37\rangle$	$\langle 62, 1, 37\rangle$
5-10	$\langle 38, 1, 61\rangle$	$\langle 37, 1, 62\rangle$	$\langle 39, 4, 57\rangle$	$\langle 43, 9, 48\rangle$	$\langle 46, 11, 43\rangle$
7-25	$\langle 32, 0, 68\rangle$	$\langle 32, 0, 68\rangle$	$\langle 33, 0, 67\rangle$	$\langle 35, 0, 65\rangle$	$\langle 21, 0, 79\rangle$

Test of KLMLean, Preferential logic P

We have tested the implementation of the calculus $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ over 600 samples. First, we have tested KLMLean over 300 sets of formulas of the KLM language $\mathcal{L}$ (sets of positive and negative conditionals). Rows in the table are labelled with a triple (Vars - Pos - Neg), representing the number of different propositional variables, the number of positive conditionals and the number of negative conditionals occurring in the initial set of formulas, respectively.

Vars - Pos - Neg	1 ms	10 ms	100 ms	1 s	2.5 s
3-5-3	$\langle 71, 0, 29\rangle$	$\langle 71, 0, 29\rangle$	$\langle 73, 0, 27\rangle$	$\langle 79, 0, 21\rangle$	$\langle 80, 0, 20\rangle$
5-10-5	$\langle 57, 0, 43\rangle$	$\langle 56, 0, 44\rangle$	$\langle 65, 0, 35\rangle$	$\langle 69, 0, 31\rangle$	$\langle 70, 0, 30\rangle$
7-15-10	$\langle 38, 0, 62\rangle$	$\langle 37, 0, 63\rangle$	$\langle 47, 0, 53\rangle$	$\langle 58, 0, 42\rangle$	$\langle 61, 0, 39\rangle$

Second, we have tested KLMLean over 300 sets of boolean combination of formulas (even conditionals), obtaining the following promising results. As for **C** and **CL**, in the following table rows are labelled with the number of propositional variables occurring and the cardinality of the sets of formulas.

Vars - Num of fml	1 ms	10 ms	100 ms	1 s	2.5 s
3-5	$\langle 43, 16, 41\rangle$	$\langle 43, 16, 41\rangle$	$\langle 44, 23, 33\rangle$	$\langle 48, 27, 25\rangle$	$\langle 48, 30, 22\rangle$
5-10	$\langle 26, 0, 74\rangle$	$\langle 26, 0, 74\rangle$	$\langle 43, 0, 57\rangle$	$\langle 68, 0, 32\rangle$	$\langle 71, 0, 29\rangle$
7-25	$\langle 32, 0, 68\rangle$	$\langle 32, 0, 68\rangle$	$\langle 34, 0, 66\rangle$	$\langle 40, 0, 60\rangle$	$\langle 44, 0, 56\rangle$

Test of KLMLean, Rational logic R

As for **P**, we have tested the implementation of the calculus $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ over 600 samples, by distinguishing 300 proofs on sets of formulas of the KLM language $\mathcal{L}$ and 300 proofs on sets of boolean combination of formulas (even conditionals). The experimental results are presented here below:

Vars - Pos - Neg	**1 ms**	**10 ms**	**100 ms**	**1 s**	**2.5 s**
3-5-3	$\langle 59, 0, 41\rangle$	$\langle 60, 0, 40\rangle$	$\langle 70, 2, 28\rangle$	$\langle 77, 3, 20\rangle$	$\langle 77, 4, 19\rangle$
5-10-5	$\langle 49, 1, 50\rangle$	$\langle 49, 1, 50\rangle$	$\langle 60, 0, 40\rangle$	$\langle 71, 0, 29\rangle$	$\langle 73, 5, 22\rangle$
7-15-10	$\langle 67, 1, 32\rangle$	$\langle 67, 1, 32\rangle$	$\langle 70, 2, 28\rangle$	$\langle 72, 5, 23\rangle$	$\langle 72, 5, 23\rangle$

Vars - Num of fml	**1 ms**	**10 ms**	**100 ms**	**1 s**	**2.5 s**
3-5	$\langle 33, 32, 35\rangle$	$\langle 32, 32, 36\rangle$	$\langle 40, 39, 21\rangle$	$\langle 43, 44, 13\rangle$	$\langle 43, 46, 11\rangle$
5-10	$\langle 19, 2, 79\rangle$	$\langle 19, 2, 79\rangle$	$\langle 35, 3, 62\rangle$	$\langle 60, 6, 34\rangle$	$\langle 63, 7, 30\rangle$
7-25	$\langle 41, 0, 59\rangle$	$\langle 41, 0, 59\rangle$	$\langle 42, 0, 58\rangle$	$\langle 44, 0, 56\rangle$	$\langle 46, 0, 54\rangle$

General Observations

From the above results, we believe that the performances of KLMLean are quite promising. It is obvious that, in general, the performances of the implementation of the version for Cumulative logic **C** are worse than the performances of other systems; indeed, the loop-checking machinery used in order to ensure termination unavoidably affects the efficiency of the resulting theorem prover. Its performances, as well as the implementation for **CL**'s (to be honest), become unacceptable with sets containing more than 20 formulas. In future research we intend to investigate if standard refinements can be applied to KLMLean in order to increase its performances.

Chapter 7

Conclusions and Future Work

This ending chapter summarizes the main results presented in this work. A discussion on possible future developments is also provided. In particular, we focus on some (nonmonotonic) extensions of Description Logics, recently introduced in order to reason about prototyipical properties and inheritance with exception and whose semantics is strongly related to the one of KLM preferential logics.

7.1 Conclusions

In this work we have provided proof methods (sequent and tableaux calculi) for logics of nonmonotonic reasoning, namely *Conditional Logics* and *Preferential Logics* (also known as KLM logics or logics of plausible reasoning). These calculi allow us to describe a decision procedure for the corresponding logics and to study their complexity. Moreover, we have implemented these calculi, obtaining efficient theorem provers for these logics. It is worth noticing that, in order to compute interesting nonmonotonic inferences, one needs to combine conditional and preferential logics with other mechanisms, such as rational closure [LM92, KLM90].

Let us summarize the main results presented in this work in more detail.

Proof Methods for Conditional Logics. We have provided a labelled calculus for minimal conditional logic CK, and its standard extensions with conditions ID, MP, CS and CEM. We have found cut-free and analytic systems for almost all studied systems, except for those presenting both MP and CEM. Basing on these calculi we have obtained a decision procedure for the respective logics. Moreover, we have been able to show that these logics are PSPACE. To the best of our knowledge, sequent calculi for these logics have not been previously studied and the complexity bound for them is new. Furthermore, we have presented a tighter space complexity bound for CK{+ID} which is based on the disjunction property of conditional formulas. We have also begun the investigation of a goal directed proof procedure for these conditional logics in the style of Miller's uniform proofs.

Proof Methods for Preferential Logics. We have presented some tableau calculi for all the KLM logical systems for nonmonotonic reasoning. We have given a tableau calculus for rational logic **R**, preferential logic **P**, loop-cumulative logic **CL**, and cumulative logic **C**. The calculi presented give a decision procedure for the respective logics. Moreover, for **R**, **P** and **CL** we have shown that we can obtain **coNP** decision procedures by refining the rules of the respective calculi. In case of **C**, we obtain a decision procedure by adding a suitable loop-checking mechanism. Our procedure gives an hyper exponential upper bound. Further investigation is needed to get a more efficient procedure. On the other hand, we are not aware of any tighter complexity bound for this logic.

Theorem Provers for Conditional and Preferential Logics. We have presented two theorem provers: CondLean, a theorem prover for conditional logics, and KLMLean, a theorem prover for KLM logics. To the best of our knowledge, these are the first theorem provers for these logics.

Both CondLean and KLMLean are SICStus Prolog implementations of the proof methods introduced in this work, namely CondLean implements sequent calculi SeqS introduced in Chapter 4, whereas KLMLean is an implementation of the tableau calculi introduced in Chapter 5.

CondLean and KLMLean are both inspired to the "lean" methodology, whose basic idea is to write short programs and exploit the power of Prolog's engine as much as possible. Both our theorem provers also comprise a graphical interface written in Java.

In this work we have also presented GoalD$\mathcal{U}$CK, a simple SICStus Prolog implementation of the goal-directed calculus $\mathcal{U}$S' introduced in Chapter 4.

7.2 Future Work

Let us conclude this work by discussing some future issues. Here again we distinguish between possible future developments for conditional logics, KLM logics and theorem proving for these logics.

Conditional Logics. The proof theoretical and complexity analysis of the systems considered in this work presents some open issues. For the systems including CEM and MP we have not been able to prove the admissibility of cut although we conjecture that it holds and we hope to prove it in future research.

For CEM the complexity bound we have found is not optimal as it is known that this logic is co-NP complete. In this work we have given uniform and modular calculi for all the logics under consideration. It might be that one can derive from our calculi an optimized calculus for CEM matching the known complexity bound. Moreover, we can use our calculi to study other logical properties such as interpolation.

We would like to study labelled sequent calculi for other conditional logics based on the selection function semantics. Among the others, the following axioms/semantic conditions are well known in the literature:

(AC) $(A \Rightarrow B) \wedge (A \Rightarrow C) \rightarrow (A \wedge C \Rightarrow B)$
If $f(w, [A]) \subseteq [B]$ then $f(w, [A \wedge B]) \subseteq f(w, [A])$

(CV) $(A \Rightarrow B) \wedge \neg(A \Rightarrow \neg C) \rightarrow (A \wedge C \Rightarrow B)$
If $f(w, [A]) \subseteq [B]$ and $f(w, [A]) \cap [C] \neq \emptyset$ then $f(w, [A \wedge C]) \subseteq [B]$

(CA) $(A \Rightarrow C) \wedge (B \Rightarrow C) \rightarrow (A \vee B \Rightarrow C)$
$f(w, [A \vee B]) \subseteq f(w, [A]) \cup f(w, [B])$

We can think of extending our calculi to these logics. We would like to have a modular proof system in the form of a sequent calculus where each semantic condition/axioms corresponds to a well-defined group of rules[1]. To this regard, it is not difficult to devise rules capturing these semantic conditions. However, a straightforward encoding of the above semantic conditions results in a non-analytic calculus where cut cannot be eliminated. The difficulty is that the selection function cannot be assumed to satisfy any *compositionality* principle: i.e. the value of $f(w, [A\#B])$ for any connective $\#$ is not a function of $f(w, [A])$ and $f(w, [B])$, at most f satisfies some constraints as the above ones. However, further research is needed to see how and whether we can capture the above semantic conditions and alike within the labelled calculus by analytic rules.

We also intend to develop goal-directed, or uniform proof, calculi for all conditional logics we have considered. This development could lead to define logic programming languages based on conditional logics.

KLM Logics. We plan to extend our calculi to the first order case. The starting point will be the analysis of first order rational logic by Friedman, Halpern and Koller in [FHK00]. In subsequent research we also intend to investigate how to find models in the alternative semantics of **P** and **R** [FH01] of a set of conditional assertions by using our tableau methods. This could be a step in order to use our tableau procedures to uniformly implement a variety of default reasoning mechanisms built upon KLM logics **P** and **R** [BDP97, BSS00, Wey03, Pea90].

Theorem Provers. In future research, we intend to extend CondLean to other systems of conditional logics. We also intend to develop goal-directed calculi for all conditional logics we have considered, in order to extend GOALD$\mathcal{U}$CK to support all of them.

We also intend to extend KLMLean to the first order case.

Moreover, we intend to increase the performances of all our theorem provers by experimenting standard refinements and heuristics.

7.2.1 Application to the Description Logics

In our recent research, we have proposed some extensions of the Description Logics with a "typicality" operator that allows us to reason about prototypical properties and inheritance with exceptions. The semantics of the typicality operator is defined by a set of postulates that are strongly related to Kraus-Lehmann-Magidor axioms of preferential logic **P**.

The family of description logics (DLs, [BCM$^+$07]) is one of the most important formalisms of knowledge representation. DLs are reminiscent of the early semantic

[1] The systems of conditional logics with axioms AC, CA and CV enjoy an alternative semantics in terms of preferential models. If one adopts this alternative semantics, one can obtain analytic proof systems as shown in the mentioned [GGOS03]. However, these proof systems take advantage of the special nature of preferential models and do not fit uniformly into the family of calculi for conditional logics with selection function semantics presented in this work.

networks and of frame-based systems. They offer two key advantages: a well-defined semantics based on first-order logic and a good trade-off between expressivity and complexity. DLs have been successfully implemented by a range of systems and they are at the base of languages for the semantic web such as OWL. A DL knowledge base (KB) comprises two components: (i) the TBox, containing the definition of concepts (and possibly roles) and a specification of inclusion relations among them, and (ii) the ABox containing instances of concepts and roles, in other words, properties and relations of individuals.

Since the primary objective of the TBox is to build a taxonomy of concepts, the need of representing prototypical properties and of reasoning about defeasible inheritance of such properties easily arises. The traditional approach is to handle defeasible inheritance by integrating some kind of nonmonotonic reasoning mechanism. However, finding a suitable nonmonotonic extension for inheritance reasoning with exceptions is far from obvious.

In [GGOP07, GGOP09a, GGOP08, GGOP09d] we have proposed an alternative approach to defeasible inheritance reasoning based on a typicality operator $\mathbf{T}$. The intended meaning is that, for any concept C, $\mathbf{T}(C)$ singles out the instances of C that are considered as "typical" or "normal". Thus an assertion like "normally students do not pay taxes" is represented by $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$.

The reasoning system should be able to infer prototypical properties as well as to ascribe defeasible properties to individuals. For instance, let the KB contain:

$\mathbf{T}(\mathit{ItalianFencer}) \sqsubseteq \neg \mathit{LovedByPeople}$
$\mathbf{T}(\mathit{ItalianFencer} \sqcap \mathit{OlympicGoldMedalist}) \sqsubseteq \mathit{LovedByPeople}$
$\mathbf{T}(\mathit{ItalianFencer} \sqcap \mathit{OlympicGoldMedalist} \sqcap \exists \mathit{TakePart}.\mathit{RealityShow}) \sqsubseteq \neg \mathit{LovedByPeople}$

corresponding to the assertions: normally an Italian fencer is not a people's favourite (fencing is not so popular in Italy...), but normally an Italian fencer who won a gold medal in an Olympic competition is a people's favourite, whereas normally an Italian Olympic gold medalist in fencing who has taken part to a reality show is not a people's favourite (because he has lost his passion in sport and his determination to obtain better and better results...). Observe that, if the same properties were expressed by ordinary inclusions, such as $\mathit{ItalianFencer} \sqsubseteq \neg \mathit{LovedByPeople}$, we would simply get that there are not Italian gold medalists in fencing and so on, thus the KB would collapse. This collapse is avoided as we do not assume that $\mathbf{T}$ is monotonic, that is to say $C \sqsubseteq D$ does not imply $\mathbf{T}(C) \sqsubseteq \mathbf{T}(D)$. Suppose next that the ABox contains the following facts about the individuals *oronzo*, *aldo* and *luca*:

1. $\mathit{ItalianFencer}(\mathit{oronzo})$
2. $\mathit{ItalianFencer}(\mathit{aldo})$, $\mathit{OlympicGoldMedalist}(\mathit{aldo})$
3. $\mathit{ItalianFencer}(\mathit{luca})$, $\mathit{OlympicGoldMedalist}(\mathit{luca}), \exists \mathit{TakePart}.\mathit{RealityShow}(\mathit{luca})$

Then the reasoning system should be able to infer the expected (defeasible) conclusions:

1. $\neg \mathit{LovedByPeople}(\mathit{oronzo})$
2. $\mathit{LovedByPeople}(\mathit{aldo})$
3. $\neg \mathit{LovedByPeople}(\mathit{luca})$

As a further step, the system should be able to infer (defeasible) properties also of individuals implicitly introduced by existential restrictions. For instance, if the ABox further contains

$\exists \mathit{HasChild}.\mathit{ItalianFencer}(\mathit{mario})$

it should conclude (defeasibly)

$$\exists HasChild.\neg LovedByPeople(mario)$$

Given the nonmonotonic character of the **T** operator, there is a difficulty with handling irrelevant information. For instance, given the KB as above, one should be able to infer as well:

$$\mathbf{T}(ItalianFencer \sqcap SlimPerson) \sqsubseteq \neg LovedByPeople$$
$$\mathbf{T}(ItalianFencer \sqcap OlympicGoldMedalist \sqcap SlimPerson) \sqsubseteq LovedByPeople$$

as *SlimPerson* is irrelevant with respect to being loved by people or not. For the same reason, the conclusion about *aldo* being a favourite of the people or not should not be influenced by the addition of $SlimPerson(aldo)$ to the ABox. We refer to this problem as the problem of Irrelevance.

In our recent research we have provided an extension of DLs with a typicality operator. Our starting point is a monotonic extension of the basic $\mathcal{ALC}$ with the **T** operator. The operator is supposed to satisfy a set of postulates that are essentially a reformulation of KLM axioms of preferential logic presented in Chapter 3, Section 3.2.2, namely, the assertion $\mathbf{T}(C) \sqsubseteq P$ is equivalent to the conditional assertion $C \mid\sim P$ of KLM preferential logic **P**. It turns out that the semantics of the typicality operator can be equivalently specified by considering a preference relation $<$ (a strict partial order) on individuals: the typical members of a concept C are just the most preferred (or "most normal") individuals of C according to the preference relation. The preference relation is the only additional ingredient that we need in our semantics. We assume that "most normal" members of a concept C always exist, whenever the concept C is non-empty. This assumption corresponds to the smoothness condition of KLM logics. Taking advantage of this semantic setting, we can give a modal interpretation to the typicality operator: for any element x, x is a typical C, i.e. it is an element of $\mathbf{T}(C)$, just in case (i) x is an element of C, and (ii) there is no y such that y is an element of C and $y < x$.

We have also defined a tableau proof procedure for $\mathcal{ALC}$ with the **T** operator. In order to formalize (ii) in the calculus, we have introduced a modality $\Box$, whose intuitive meaning is that x is an element of $\Box C$ if, for every $y < x$, we have that y is an element of C. The basic idea of the calculus is very similar to the one of the tableau calculi introduced in Chapter 5 for KLM logics, that is to say to interpret the preference relation $<$ as an accessibility relation. By the smoothness condition, it turns out that the modality $\Box$ has the properties of Gödel-Löb modal logic of provability G, exactly as in the calculi for KLM logics.

From a computational viewpoint, in [GGOP09a] we have shown that the extension of $\mathcal{ALC}$ with the **T** operator is decidable and we have provided an EXPTIME complexity upper bound. Since reasoning in $\mathcal{ALC}$ alone with arbitrary TBox has already the same complexity, we can conclude that the extension by **T** is essentially inexpensive.

From a knowledge representation viewpoint, however, the extension we have proposed is not enough to perform inheritance reasoning of the kind described above. We need a way of inferring defeasible properties of individuals and a way of handling Irrelevance. In [GGOP07, GGOP09a] we have defined a *completion* of an ABox: the idea is that each individual is assumed to be a typical member of the most specific concept to which it belongs. Such a completion allows to perform inferences as 1.,2.,3. above. In [GGOP08] we have strengthened the semantics of $\mathcal{ALC} + \mathbf{T}$ by proposing a *minimal model semantics*. Intuitively, the idea is to restrict our consideration to models that maximise typical instances of a concept. The first proposal is computationally easy, but it presents some difficulties. The second proposal is computationally more expensive, but it is more powerful for inheritance reasoning.

We have also extended our approach based on the typicality operator $\mathbf{T}$ to some *light-weight* DLs, namely the logics of the well known $\mathcal{EL}$ family. The logics of the $\mathcal{EL}$ family allow for conjunction ($\sqcap$) and existential restriction ($\exists R.C$). Despite their relatively low expressivity, these logics are relevant for several applications, in particular in the bio-medical domain. In [GGOP09c, GGOP09e] we have introduced the logic $\mathcal{EL}^{+^{\perp}}\mathbf{T}$, then we have shown that the problem of deciding entailment in $\mathcal{EL}^{+^{\perp}}\mathbf{T}$ is in co-NP.

Bibliography

[AA00] O. Arieli and A. Avron. General patterns for nonmonotonic reasoning: From basic entailments to plausible relations. *Logic Journal of the IGPL*, 8(2):119–148, 2000.

[AG98] A. Artosi and G. Governatori. A tableaux methodology for deontic conditional logics. In *DEON'98, 4^{th} International Workshop on Deontic Logic in Computer Science*, pages 65–81, Bologna, 1998. CIRFID.

[AGR02] A. Artosi, G. Governatori, and A. Rotolo. Labelled tableaux for non-monotonic reasoning: Cumulative consequence relations. *Journal of Logic and Computation*, 12(6):1027–1060, 2002.

[Avr96] A. Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In Wilfrid Hodges, Martin Hyland, Charles Steinhorn, and John Truss, editors, *Logic: from foundations to applications.*, pages 1–32. Oxford University Press, New York, 1996.

[BCM$^+$07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2007.

[BDP97] S. Benferhat, D. Dubois, and H. Prade. Nonmonotonic reasoning, conditional objects and possibility theory. *Artificial Intelligence*, 92(1-2):259–276, 1997.

[Bel91] J. Bell. Pragmatic logics. In R. Fikes and E. Sandewall, editors, *Proceedings of Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference KR'91*, pages 50–60, Cambridge, Massachusetts, USA, 1991.

[BG97] B. Beckert and R. Goré. Free variable tableaux for propositional modal logics. *In Proceedings of TABLEAUX 1997 (Automated Reasoning with Analytic Tableaux and Related Methods), volume 1227 of LNAI, Springer-Verlag*, pages 91–106, 1997.

[BG01] B. Beckert and R. Goré. Free-variable tableaux for propositional modal logics. *Studia Logica*, 69(1):59–96, 2001.

[Bil93] D. Billington. Defeasible logic is stable. *Journal of Logic and Computation*, 3(4):379–400, 1993.

[Bou94] C. Boutilier. Conditional logics of normality: a modal approach. *Artificial Intelligence*, 68(1):87–154, 1994.

[BP95] B. Beckert and J. Posegga. leantap: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.

[BP96] B. Beckert and J. Posegga. Logic programming as a basis for lean automated deduction. *Journal of Logic Programming*, 28(3):231–236, 1996.

[BSS00] S. Benferhat, A. Saffiotti, and P. Smets. Belief functions and default reasoning. *Artificial Intelligence*, 122(1-2):1–69, 2000.

[Bur81] J. P. Burgess. Quick completeness proofs for some logics of conditionals. *Notre Dame Journal of Formal Logic*, 22:76–84, 1981.

[CdC95] G. Crocco and L. Fariñas del Cerro. Structure, consequence relation and logic, volume 4. *In D. M. Gabbay (ed.), What is a Logical System, Oxford University Press*, pages 239–259, 1995.

[Che75] B. F. Chellas. Basic conditional logics. *Journal of Philosophical Logic*, 4:133–153, 1975.

[CL92] G. Crocco and P. Lamarre. On the connection between non-monotonic inference systems and conditional logics. In B. Nebel and E. Sandewall, editors, *Proceedings of Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference KR'92*, pages 565–571, 1992.

[Cla78] K. L. Clark. Negation as failure. *In H. Gallaire and J. Minker, editors,* Logics and Data Bases*, Plenum Press*, pages 293–322, 1978.

[CM99] T. Costello and J. McCarthy. Useful counterfactuals. *ETAI (Electronic Transactions on Artificial Intelligence)*, 3:Section A, 1999.

[Del87] J. P. Delgrande. A first-order conditional logic for prototypical properties. *Artificial Intelligence*, 33(1):105–130, 1987.

[DFP03] D. Dubois, H. Fargier, and P. Perny. Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach. *Art. Int.*, 148(1-2):219–260, 2003.

[DFPP02] D. Dubois, H. Fargier, P. Perny, and H. Prade. Qualitative decision theory: from savages axioms to nonmonotonic reasoning. *Journal of the ACM*, 49(4):455–495, 2002.

[DG90] J. P. Delgrande and C. Groeneboer. A general approach for determining the validity of commonsense assertions using conditional logics. *International Journal of Intelligent Systems*, 5(5):505–520, 1990.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22, 1979.

[dS83] H. C. M. de Swart. A gentzen- or beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics vc and vcs. *Journal of Symbolic Logic*, 48(1):1–20, 1983.

[FH94] N. Friedman and J. Halpern. On the complexity of conditional logics. *In Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference, KR 1994*, pages 202–213, 1994.

[FH01] N. Friedman and J. Y. Halpern. Plausibility measures and default reasoning. *Journal of the ACM*, 48(4):648–685, 2001.

[FHK00] N. Friedman, J. Y. Halpern, and D. Koller. First-order conditional logic for default reasoning revisited. *ACM Transactions on Computational Logics (TOCL)*, 1(2):175–207, 2000.

[Fit98] M. Fitting. leantap revisited. *Journal of Logic and Computation*, 8(1):33–47, 1998.

[Gab85] D. Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. *Logics and models of concurrent systems, Springer*, pages 439–457, 1985.

[Gab96] D. M. Gabbay. Labelled deductive systems (vol i). *Oxford Logic Guides, Oxford University Press*, 1996.

[Gar88] P. Gardenförs. *Knowledge in Flux.* MIT Press, 1988.

[Gen92] I. P. Gent. A sequent or tableaux-style system for lewis's counterfactual logic vc. *Notre Dame Journal of Formal Logic*, 33(3):369–382, 1992.

[GGM+00] D. M. Gabbay, L. Giordano, A. Martelli, N. Olivetti, and M. L. Sapino. Conditional reasoning in logic programming. *Journal of Logic Programming*, 44(1-3):37–74, 2000.

[GGO02] L. Giordano, V. Gliozzi, and N. Olivetti. Iterated belief revision and conditional logic. *Studia Logica*, 70(1):23–47, 2002.

[GGO05] L. Giordano, V. Gliozzi, and N. Olivetti. Weak agm postulates and strong ramsey test: a logical formalization. *Artificial Intelligence*, 168(1-2):1–37, 2005.

[GGOP05a] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. A Tableaux Calculus for KLM Preferential and Cumulative Logics. In Bernhard Beckert, editor, *Position Papers and Tutorial Description of TABLEAUX 2005*, volume 12/2005, pages 11–26, Koblenz, Germany, September 2005. Fachberichte INFORMATIK.

[GGOP05b] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux for KLM Preferential and Cumulative Logics. In Geoff Sutcliffe and Andrei Voronkov, editors, *Proceedings of LPAR 2005 (12th Conference on Logic for Programming, Artificial Intelligence, and Reasoning)*, volume 3835 of *LNAI*, pages 666–681, Montego Bay, Jamaica, December 2005. Springer-Verlag.

[GGOP06a] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux Calculi for KLM Rational Logic R. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Proceedings of JELIA 2006 (10th European Conference on Logics in Artificial Intelligence)*, volume 4160 of *LNAI*, pages 190–202, Liverpool, England, September 2006. Springer-Verlag.

[GGOP06b] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Automated Deduction for Logics of Default Reasoning. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proceedings of ECAI 2006 (17th European Conference on Artificial Intelligence)*, pages 757–758, Riva del Garda, Italy, August-September 2006. IOS Press.

[GGOP07] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Preferential Description Logics. In Nachum Dershowitz and Andrei Voronkov, editors, *Proceedings of LPAR 2007 (14th Conference on Logic for Programming, Artificial Intelligence, and Reasoning)*, volume 4790 of *LNAI*, pages 257–272, Yerevan, Armenia, October 2007. Springer-Verlag.

[GGOP08] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Reasoning About Typicality in Preferential Description Logics. In S. Hólldobler, C. Lutz, and H. Wansing, editors, *Proceedings of JELIA 2008 (11th European Conference on Logics in Artificial Intelligence)*, volume 5293 of *LNAI*, pages 192–205, Dresden, Germany, September 2008. Springer-Verlag.

[GGOP09a] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. ALC+**T**: a preferential extension of description logics. *Fundamenta Informaticae*, 96:1–32, 2009.

[GGOP09b] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux Calculi for KLM Logics of Nonmonotonic Reasoning. *ACM Transactions on Computational Logics (TOCL)*, 10(3), 2009.

[GGOP09c] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Prototypical reasoning with low complexity description logics: preliminary results. In F. Lin, T. Schaub, and E. Erdem, editors, *Proceedings of LPNMR 2009 (10th International Conference on Logic Programming and Nonmonotonic Reasoning)*, volume 5753 of *LNAI*, pages 430–436, Potsdam, Germany, September 2009. Springer-Verlag.

[GGOP09d] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Reasoning about typicality in $\mathcal{ALC}$ and $\mathcal{EL}$. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, volume CEUR Workshop Proceedings. Vol. 477, pages 24/1–24/13, Oxford, United Kindgom, July 27-30 2009.

[GGOP09e] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Reasoning about typicality with low complexity description logics: the logic $\mathcal{EL}^{+^\perp}\mathbf{T}$. In R. Serra and R. Cucchiara, editors, *Proceedings of AI*IA 2009 (XI Conference of the Italian Association for Artificial Intelligence)*, volume 5883 of *LNAI*, pages 62–71, Reggio Emilia, Italy, December 9-12 2009. Springer-Verlag.

[GGOS03] L. Giordano, V. Gliozzi, N. Olivetti, and C. Schwind. Tableau calculi for preference-based conditional logics. In Marta Cialdea Meyer and Fiora Pirri, editors, *Proceedings of TABLEAUX 2003 (Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 2796 of *LNAI*, pages 81–101, Roma, Italy, September 2003. Springer.

[GGOS05] L. Giordano, V. Gliozzi, N. Olivetti, and C.B. Schwind. Extensions of tableau calculi for preference-based conditional logics. In Holger Schlingloff, editor, *Proceedings of the 4th International Workshop on Methods for Modalities (M4M-4)*, pages 220–234, Fraunhofer Institute FIRST, Berlin, Germany, December 2005. Informatik-Bericht 194.

[GGP07] L. Giordano, V. Gliozzi, and G. L. Pozzato. KLMLean 2.0: A Theorem Prover for KLM Logics of Nonmonotonic Reasoning. In N. Olivetti, editor, *Proceedings of TABLEAUX 2007 (16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 4548 of *LNAI*, pages 238–244, Aix En Provence, France, July 2007. Springer-Verlag.

[Gin86] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, 1986.

[GO00] D. M. Gabbay and N. Olivetti. *Goal-directed Proof Theory.* Kluwer Academic Publishers, 2000.

[Gor99] R. Goré. Tableau methods for modal and temporal logics. *In Handbook of Tableau Methods, Kluwer Academic Publishers*, pages 297–396, 1999.

[GP98] D. Galles and J. Pearl. An axiomatic characterization of causal counterfactuals. *Foundation of Science*, 3(1):151–182, 1998.

[Gra98] G. Grahne. Updates and counterfactuals. *Journal of Logic and Computation*, 8(1):87–117, 1998.

[GS04] L. Giordano and C. Schwind. Conditional logic of actions and causation. *Artificial Intelligence*, 157(1-2):239–279, 2004.

[HC84] G.E. Hughes and M.J. Cresswell. *A Companion to Modal Logic.* Methuen, 1984.

[HSZ96] A. Heuerding, M. Seyfried, and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In *Proceedings of TABLEAUX 1996, volume 1071 of LNAI, Springer*, pages 210–225, 1996.

[Hud93] J. Hudelmaier. An $\mathcal{O}(n \ log \ n)$-space decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75, 1993.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.

[KS91] H. Katsuno and K. Sato. A unified view of consequence relation, belief revision and conditional logic. In *Proceedings of IJCAI'91*, pages 406–412, 1991.

[Lam92] P. Lamarre. A promenade from monotonicity to non-monotonicity following a theorem prover. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Proceedings of Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference KR'92*, pages 572–580, Cambridge, Massachusetts, USA, 1992.

[Lew73] D. Lewis. Counterfactuals. *Basil Blackwell Ltd*, 1973.

[LM92] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.

[Mak03] D. Makinson. Bridges between classical and nonmonotonic logic. *Logic Journal of the IGPL*, 11(1):69–96, 2003.

[Mak05] D. Makinson. *Bridges from Classical to Nonmonotonic logic.* London: King's College Publications. Series: Texts in Computing, vol 5, 2005.

[McC80] J. McCarthy. Circumscription – a form of non monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[MD80] D. McDermott and J. Doyle. Non-monotonic logic i. *Artificial Intelligence*, 13:41–72, 1980.

[MH94] D. Miller and J. S. Hodas. Logic programming in a fragment of intuitionistic linear logic. *Journal of Information and Computation*, 110(2):327–365, 1994.

[MNPS91] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *In Annal of Pure and Applied Logic*, 51(1-2):125–157, 1991.

[Moo84] R.C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the 1st AAAI Workshop on Nonmonotonic Reasoning*, pages 344–354, New Palz, New York, 1984.

[Moo85] R.C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Nut80] D. Nute. Topics in conditional logic. *Reidel, Dordrecht*, 1980.

[Nut84] D. Nute. Conditional logic. *Handbook of Philosophical Logic, volume 2, Extensions of Classical Logic, ed. by D. M. Gabbay, and F. Guenthner*, 1984.

[Obe01] N. Obeid. Model-based diagnosis and conditional logic. *Applied Intelligence*, 14:213–230, 2001.

[OP03] N. Olivetti and G. L. Pozzato. CondLean: A Theorem Prover for Conditional Logics. In Marta Cialdea Meyer and Fiora Pirri, editors, *Proceedings of TABLEAUX 2003 (Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 2796 of *LNAI*, pages 264–270, Roma, Italy, September 2003. Springer.

[OP05a] N. Olivetti and G. L. Pozzato. CondLean 2.0: an Efficient Theorem Prover for Standard Conditional Logics. In Jean-Yves Beziau and Alexandre Costa-Leite, editors, *Handbook of the 1st World Congress and School on Universal Logic*, pages 89–90. Montreux, Switzerland, 2005.

[OP05b] N. Olivetti and G. L. Pozzato. CondLean 3.0: Improving Condlean for Stronger Conditional Logics. In Bernhard Beckert, editor, *Proceedings of TABLEAUX 2005 (Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 3702 of *LNAI*, pages 328–332, Koblenz, Germany, September 2005. Springer-Verlag.

[OP05c] N. Olivetti and G. L. Pozzato. KLMLean 1.0: a Theorem Prover for Logics of Default Reasoning. In Holger Schlingloff, editor, *Proceedings of the 4th International Workshop on Methods for Modalities (M4M-4)*, pages 235–245, Fraunhofer Institute FIRST, Berlin, Germany, December 2005. Informatik-Bericht 194.

[OP07] N. Olivetti and G. L. Pozzato. Automated Reasoning for Conditional Logics: the Theorem Prover CondLean 3.1. In Jean-Yves Beziau and Alexandre Costa-Leite, editors, *Perspectives on Universal Logic*, pages 395–415. Polimetrica International Scientific Publisher, Monza, Italy, 2007.

[OP08] N. Olivetti and G. L. Pozzato. Theorem Proving for Conditional Logics: CondLean and GoalDuck. *Journal of Applied Non-Classical Logics (JANCL)*, 18(4):427–473, 2008.

[OPS05] N. Olivetti, G. L. Pozzato, and C. B. Schwind. A sequent calculus and a theorem prover for standard conditional logics: Extended version. *Technical Report 87/05, Dip. di Informatica, Università degli Studi di Torino, available at http://www.di.unito.it/~argo/biblio/publications.php?author=Pozzato*, 2005.

[OPS07] N. Olivetti, G. L. Pozzato, and C. B. Schwind. A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. *ACM Transactions on Computational Logics (TOCL)*, 8(4), 2007.

[OS00] N. Olivetti and C. B. Schwind. Analytic tableaux for conditional logics. *Technical Report, University of Torino*, 2000.

[otSIoCS06] Intelligent Systems Laboratory of the Swedish Institute of Computer Science. Sicstus prolog users manual. *http://www.sics.se/isl/sicstuswww/site/documentation.html*, 2006.

[Pea90] J. Pearl. System z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 121–135, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[Pea99] J. Pearl. Reasoning with cause and effect. In T. Dean, editor, *Proceedings of IJCAI 1999 (16th International Joint Conference on Artificial Intelligence)*, pages 1437–1449, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Pea00] J. Pearl. *Causality: Model, Reasoning and Inference.* Cambridge University Press, 2000.

[PH94] D. J. Pym and J. Harland. A uniform proof-theoretic investigation of linear logic programming. *Journal of Logic and Computation*, 4(2):175–207, 1994.

[Poz03] G. L. Pozzato. Deduzione automatica per logiche condizionali: Analisi e sviluppo di un theorem prover. *Tesi di laurea, Informatica, Università di Torino. In Italian, download at* `http://www.di.unito.it/~pozzato/tesiPozzato.html`, 2003.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Sch99] C. B. Schwind. Causality in action theories. *Electronic Transactions on Artificial Intelligence (ETAI)*, 3(A):27–50, 1999.

[Sho87a] Y. Shoham. *Reasoning about change.* The MIT Press, 1987.

[Sho87b] Y. Shoham. A semantical approach to nonmonotonic logics. *In Proceedings of Logics in Computer Science*, pages 275–279, 1987.

[Smu68] R.M. Smullyan. *First-Order Logic.* Springer-Verlag, Berlin, 1968.

[Sta68] R. Stalnaker. A theory of conditionals. *In N. Rescher (ed.), Studies in Logical Theory, American Philosophical Quarterly, Monograph Series no.2, Blackwell, Oxford*, pages 98–112, 1968.

[Tou86] D. S. Touretzky. *The mathematics of inheritance systems.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986.

[Var88] M. Y. Vardi, editor. *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning about Knowledge, Pacific Grove, CA, March 1988.* Morgan Kaufmann, 1988.

[Vig00] L. Viganò. Labelled non-classical logics. *Kluwer Academic Publishers, Dordrecht*, 2000.

[Wey03] E. Weydert. System jlz - rational default reasoning by minimal ranking constructions. *Journal of Applied Logic*, 1(3-4):273–308, 2003.