Lecture Notes in Artificial Intelligence 4696

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Hans-Dieter Burkhard Gabriela Lindemann
Rineke Verbrugge László Z. Varga (Eds.)

Multi-Agent Systems and Applications V

5th International
Central and Eastern European Conference
on Multi-Agent Systems, CEEMAS 2007
Leipzig, Germany, September 25-27, 2007
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Hans-Dieter Burkhard
Gabriela Lindemann
Humboldt-Universität zu Berlin
Institut für Informatik, LFG Künstliche Intelligenz
Rudower Chaussee 25, 12489 Berlin, Germany
E-mail: {hdb, lindemann}@informatik.hu-berlin.de

Rineke Verbrugge
University of Groningen
Institute of Artificial Intelligence
Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands
E-mail: rineke@ai.rug.nl

László Z. Varga
Computer and Automation Research Institute
Kende u. 13-17, 1111, Budapest, Hungary
E-mail: laszlo.varga@sztaki.hu

Library of Congress Control Number: 2007935205

CR Subject Classification (1998): I.2.11, I.2, C.2.4, D.2, H.5.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-75253-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75253-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12165964 06/3180 5 4 3 2 1 0

Preface

The aim of the CEEMAS conference series is to provide a biennial forum for the presentation of multi-agent research and development results. With its particular geographical orientation towards Central and Eastern Europe, CEEMAS has become an internationally recognized event with participants from all over the world. After the successful CEEMAS conferences in St. Petersburg (1999), Cracow (2001), Prague (2003) and Budapest (2005), the CEEMAS 2007 conference took place in Leipzig. The Program Committee of the conference series consists of established researchers from the region and renowned international colleagues, showing the prominent rank of CEEMAS among the leading events in multi-agent systems.

In the very competitive field of agent-oriented conferences and workshops, (such as AAMAS, EUMAS, CIA, MATES) CEEMAS is special in trying to bridge the gap between applied research achievements and theoretical research activities. The ambition of CEEMAS is to provide a forum for presenting theoretical research with an evident application potential, implemented application prototypes and their properties, as well as industrial case studies of successful (or unsuccessful) agent technology deployments. This is why the CEEMAS proceedings provide a collection of research and application papers. The technical research paper section of the proceedings (see pages 1–290) contains pure research papers as well as research results in application settings. The goal is to demonstrate the real-life value and commercial reality of multi-agent systems as well as to foster the communication between academia and industry in this field.

CEEMAS is also very special and unique because it is constantly contributing to building an agent research community. The Program Committee has decided to create a special collection of short papers that provide an opportunity to present ongoing research work with a potential of mature and higher impact research results in the near future. This allows researchers to expose their work for constructive criticism and discuss their projects with other experts in an early phase of their research. On the other hand this provides the audience with fresh, innovative and highly motivating ideas that may deserve further investigation.

The topics of the CEEMAS proceedings cover a wide range of areas such as: abstract and specific agent architectures, methods and modeling approaches for agent-oriented software engineering, agent communication and protocols. CEEMAS also features papers on applications from the field of healthcare, traffic management, learning, software development, game, production management, trade and negotiations.

We received 84 submissions, and each paper was reviewed by at least two independent reviewers or the General Co-chairs. Out of the submitted papers, 29 were accepted as full research papers and 17 as short poster papers.

Many individuals and institutions supported the organization of this conference and made CEEMAS 2007 a high-quality event. Our special thanks go first to the authors and invited speakers for their invaluable and strenuous work and contribution. Also, the work of the Program Committee members who accepted the heavy load of the review of a large number of contributions is gratefully acknowledged. We are especially thankful to the University of Leipzig for their organizational activities.

As a result, the present collection of papers provides a valuable resource for researchers in the field of multi-agent systems and open distributed systems in general.

July 2007

Hans-Dieter Burkhard
Gabriela Lindemann
László Z. Varga
Rineke Verbrugge

Organization

CEEMAS 2007 was co-located with the Software Agents and Services for Business, Research, and E-Sciences (SABRE) set of conferences organized by the Management Information Systems Institute and the Institute for Applied Informatics (InfAI) of the University of Leipzig.

CEEMAS Steering Committee

Barbara Dunin-Keplicz, Poland
Vladimir Gorodetski, Russia
Michael Luck, UK
Vladimír Mařík, Czech Republic
Jörg Müller, Germany
Edward Nawarecki, Poland
Michal Pěchouček, Czech Republic
Paolo Petta, Austria
László Z. Varga, Hungary

CEEMAS 2007 General Co-chairs

Hans-Dieter Burkhard, Germany
Gabriela Lindemann, Germany
László Z. Varga, Hungary
Rineke Verbrugge, The Netherlands

CEEMAS 2007 Program Committee

Roberto A. Flores, Canada
Magnus Boman, Sweden
Luis Botelho, Portugal
Krzysztof Cetnarowicz, Poland
Yves Demazeau, France
Frank Dignum, The Netherlands
Grzegorz Dobrowolski, Poland
Danail Dochev, Bulgaria
Enrico Gerding, UK
Marie-Pierre Gleizes, France
Chihab Hanachi, France
Karin Hummel, Austria
Toru Ishida, Japan
Oleg Karsaev, Russia
Franziska Klügl, Germany
James Lawton, USA
Gabriela Lindemann, Germany
Micheal Luck, UK
John-Jules Meyer, The Netherlands
László Monostori, Hungary
Pablo Noriega, Spain
Andrea Omicini, Italy
Marek Paralic, Slovakia
Agostino Poggi, Italy
Andrzej Skowron, Poland
Ingo Timm, Germany

Robert Tolksdorf, Germany
Paul Valckenaers, Belgium
Wiebe van der Hoek, UK
József Váncza, Hungary
Herbert Wiklicky, UK

Additional Reviewers

Ralf Berger
Xavier Clerc
Ludivine Crepin
Joris Deguet
Pierre Glize
Daniel Goehring
Helmut Hlavacs
Marc-Philippe Huget
Matthias Jüngel
Jussi Karlgren

Table of Contents

Full Papers

Short Papers

A Multi-agent Approach for Range Image Segmentation

Smaine Mazouzi[1], Zahia Guessoum[2], Fabien Michel[1], and Mohamed Batouche[3]

[1] MODECO-CReSTIC, Université de Reims, B.P. 1035, 51687, Reims, France
{mazouzi,fmichel}@leri.univ-reims.fr
[2] LIP6, Université de Paris 6, 104, av. du Président Kennedy, 75016, Paris, France
zahia.guessoum@lip6.fr
[3] Département d'informatique, Université de Constantine, 25000, Algérie
batouche@wissal.dz

Abstract. This paper presents and evaluates a multi-agent approach for range image segmentation. A set of reactive and autonomous agents perform a collective segmentation by partitioning a range image in its different planar regions. The agents move over the image and perform cooperative and competitive actions on the pixels, allowing a robust region extraction, and an accurate edge detection. An artificial potential field, created around the pixels of interest, ensures the agent coordination. It allows the agents to concentrate their actions around the edges and the noise regions. The experimental results show the potential of the proposed approach for scene understanding in range images, regarding both segmentation efficiency, and detection accuracy.

Keywords: Image segmentation, Multi-agent systems, Range image, Agent coordination, Artificial potential field.

1 Introduction

Image segmentation consists in assigning the pixels of an image to homogenous and disjoint subsets, providing a compact and convenient description of the image. In range imagery, segmentation algorithms can be divided into two dual categories: edge-based segmentation algorithms and region-based segmentation algorithms. In the first category, pixels that correspond to discontinuities in depth or in surface normals are selected, chained and then used to delimit the regions in the image [6,7]. Region-based methods use geometrical surface descriptors to group pixels, with the same proprieties, in disjoint regions [8,10,3,1]. In both categories, algorithms must deal with noisy and uncertain data.

To overcome this difficulty, some authors have proposed multi-agent systems for 2-D image segmentation. Solutions provided by such systems inherit the advantages of the agent-oriented approach for collective problem solving. In such systems a single agent has a limited perception and limited capabilities, and it is not designed to solve an entire problem. Agents cooperate thus in order to provide a collective solution. Contrary to conventional systems, solutions in multi-agent systems emerge from the collective action of interactive agents [9].

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 1–10, 2007.

In this paper, a new multi-agent approach for range image segmentation is presented and discussed. It consists in the use of reactive agents, which move over the image, and act on the visited pixels. While moving over the image, an agent adapts to the planar region on which it moves, and memorizes its proprieties. At the boundaries between regions, the agents will be in competition to align the pixels of the boundaries to their respective regions. The resulting alternative alignment of the boundary pixels preserves the region boundaries against smoothing. Noise regions, which are characterized by small sizes or by aberrant depths (outliers), prevent agents from adapting. So, the pixels on their borders are continuously aligned to the true regions that surround them. After several iterations these regions will entirely disappear.

This work aims to overcome the difficulty related to the local perception around the processed pixel. A pixel is therefore processed according to both its neighborhood, and the agents that visit this pixel. An agent acts on the pixels with more certainty, acquired from its move on large areas on the image regions. The combination of the global information memorized within the agent, and the local information of the image, provides more reliable decisions. We show in this work that despite the simplicity of the model used to represent the image surface, the obtained results are better than those provided by conventional approaches. We believe that interactions between agents provide an alternative way for image segmentation to that of approaches based on complicated and costly models. Extensive experiments have been performed using real images from the ABW database [5]. The obtained results show the high potential of the proposed approach for an efficient and accurate segmentation of range images.

The remainder of the paper is organized as follows: In Section 2, we review some agent-based approaches for image segmentation. Section 3 is devoted to the proposed approach. It describes the behavior of the agents, and shows the underlying collective mechanism to deal with the edge detection and the noise removal. The experimental results are introduced in Section 4, in which we discuss the parameter selection, and we analyze and comment the obtained results. Finally, a conclusion summarizes our contribution.

2 Related Work

Several agent-based systems have been proposed to deal with various problems in image analysis and object recognition. In this review we consider only some works that have addressed a solution in image segmentation.

Liu et al. [11] introduce a reactive agent-based system for brain MRI segmentation. Four types of agents are used to label the pixels of the image according to their membership grade to the various regions. In this system, the agents neither interact directly between them nor act on the image. Their actions depend only on their local perception. Nevertheless, each agent is created so that it becomes more likely to meet more homogenous pixels. For the same type of images, Richard et al. [12] propose a hierarchical architecture of situated and cooperative agents. Several types of agents were used. Interaction between agents

in the various hierarchic levels has allowed to deal with the control over the low-level segmentation tasks. However, the system was specially optimized to brain MRI segmentation. The two previous systems can provide correct results because region characteristics are regular in the various brain anatomic parts. In addition, most of the edges in such images are jump edges (at discontinuities of image data), which are easy to detect, compared to roof or smooth edges (edges respectively at normal or curvature discontinuities).

Based on a cognitive architecture, Bovenkamp et al. [2] have developed a multi-agent system for Intra Vascular Ultra Sound (IVUS) image segmentation. They aim to elaborate a high knowledge-based control over the low-level image processing algorithms. In this system, an agent is assigned to every expected object in the image. In this work, the problem of the control over the segmentation algorithms seems to be well resolved. However, no agent or even behavior has been proposed to deal with uncertain and noisy data.

The proposed agent-based systems for image segmentation are specific to image contents. Following a supervised approach, these systems segment images in known and previously expected regions. The system proposed in this paper claims to be general and unsupervised. It aims to segment an image into its different regions by using some invariant surface proprieties. The adaptive and competitive behavior of the agents allows overcoming the constraint related to the restriction of the treatments to the local neighborhood of the pixels.

3 Multi-agent Range Image Segmentation

A range image is a discretized two-dimensional array where at each pixel (x, y) is recorded the distance $Z(x, y)$ between the range finder and the corresponding point of the scene. A new image Z^*, called plane image is derived from the range image. Each pixel (x, y) of the new image records the tangent plane to the surface at (x, y). The tasks performed by the agents on the plane image are based on the comparison of planes. So, we consider that two planes $ax + by + cz = d$ and $a'x + b'y + c'z = d'$ are equal if they have, according to given thresholds (Tr_θ, Tr_D), the same orientation, and the same distance to the coordinate origin. Tr_θ and Tr_D are respectively the angle and the distance thresholds.

The plane image Z^* is considered as the environment in which the agents are initialized at random positions. An agent checks if it is situated on a planar region, and adapts to this region if it is planar, by memorizing its plane equation. Next, the agent performs actions, which depend on both its state and the state of the pixel on which it is located. At each time t, an agent is characterized by its position (x_t, y_t) over the image, and by its ability A_t to act on the encountered pixels. At the beginning of the process, all the agents are unable to alter any pixel of the image. After having been adapted to a planar region, an agent becomes able to modify the first encountered pixel that not belongs to the current region (A_t=true). When an agent alters a pixel, it loses its alteration ability (A_t=false) and starts again searching for a new planar region. An agent having modified a pixel records in an appropriate two-dimensional array I, at (x_t, y_t) the last state

of the visited pixel: $I(x_t, y_t) \in \{\text{smoothed, aligned, unchanged}\}$. We show next, that this simple behavior of the agents allows both the detection of the image edges, and the removal of the noise regions.

3.1 Agent Behavior

An agent adapts to the region of the image on which it is moving by computing and memorizing the proprieties of this region, and by adopting the suited behavior to the local image data. An agent can be in one of the following situations : searching for a planar region, moving on a planar region, aligning images pixels.

After its creation, an agent randomly moves within the image and searches for a planar region around its current position. The agent uses a region seed formed by the last L visited pixels. L is called the adaptation path-length. It represents the confidence degree that the agent is situated within a planar region. So, the agent considers that it is within a planar region if the pixels of the seed form a planar surface. The agent memorizes the new region and considers it as its current planar region. It becomes then able to alter the first encountered pixel that does not belong to its new region (A_t=true).

While moving inside a planar region, an agent smoothes the pixel on which it is located, by updating the equations of both the memorized plane and the plane at the position of the pixel. This is done by replacing the two equations by their weighted average. Let (a, b, c, d) and (a', b', c', d') be the parameters respectively of the plane at the current pixel, and the memorized plane. Resulting parameters of the weighted average plane are obtained as follows:

$$(a'', b'', c'', d'') = \frac{1}{1+l}(a + la', b + lb', c + lc', d + ld') \tag{1}$$

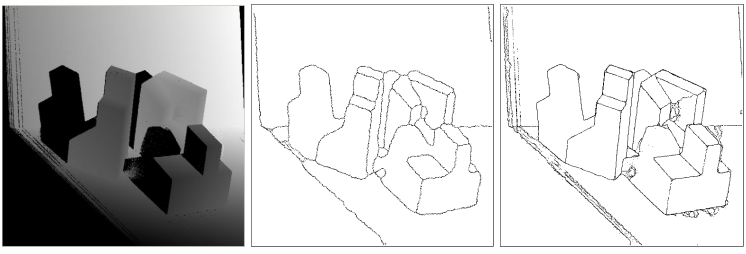
where l is the length of the path crossed by the agent on the planar region.

When an agent meets a pixel of interest (not belonging to its current planar region, i.e. noise pixels or edge pixels), the pixel is partially aligned to the planar region on which the agent moves. The parameters (a'', b'', c'', d'') of the new plane equation at the pixel position are obtained by linear combination of the current parameters (a, b, c, d) and the parameters of the memorized plane equation (a', b', c', d'):

$$(a'', b'', c'', d'') = \frac{1}{1+\xi}(a + \xi a', b + \xi b', c + \xi c', d + \xi d') \tag{2}$$

where ξ is the alteration strength.

The agent loses then its alteration ability (A_t=false) and starts again to search for a new planar region. The alteration strength ξ is a critical parameter, which affects the quality of the results and the time of computation. Indeed, high values of ξ lead to a fast region detection. However, the resulting region boundaries are distorted and badly localized (Fig. 1b). Low values of ξ result in a slow detection, but region boundaries in this case, are well detected and correctly localized (Fig. 1c). In order to speed up the segmentation process, without edge distortion, an agent chooses the alteration strength among ξ_{min} and ξ_{max} according to the information recorded by other agents in the array I. So, an agent assumes that



(a) (b) (c)

Fig. 1. The impact of the alteration strength on the segmentation results: (a) Range image (abw.test.8); (b) Segmentation results with $\xi_{min} = \xi_{max} = 4$ at t=2500; (c) Segmentation results with $\xi_{min} = 0.3$ and $\xi_{max} = 5$ at t=13000

the current planar region is adjacent to a noise region, and thus uses ξ_{max} as alteration strength, if the number of "unchanged" pixels (situated within a noise region) around the agent is greater than a given threshold (fixed to 3 in our experimentations). Indeed, pixels labeled "unchanged" in the adjacent region mean that this latter is a noise region for which agents have not adapted and consequently have not smoothed its pixels. Otherwise, the agent assumes that the current planar region is adjacent to another planar one, where other agents have labeled the pixels as "smoothed" or "aligned". In this case, the agent uses the alteration strength ξ_{min}.

3.2 Agent Coordination by Artificial Potential Field

To endow agents with a self-organization mechanism, an artificial electrostatic-like potential field is used. It is created and updated around the aligned pixels. It allows agents to be gathered around pixels of region boundaries, and concentrate their actions at these pixels. Contrary to other works, where the potential field is created at known positions of objects (goals and obstacles) [4,13], the potential field in our case results from the interaction of agents with the objects in the environment (pixels). The intensity $\Psi(x, y)$ of the potential field at position (x, y) created by a set of P pixels beforehand aligned $\{(x_i, y_i), i = 1..P \wedge I(x_i, y_i)$=aligned$\}$ is given by:

$$\Psi(x,y) = \sum_{i=1}^{P} \frac{k}{\sqrt{(x-x_i)^2 + (y-y_i)^2}}, k \in R^+ \tag{3}$$

where k is the constant of the electrostatic force, set to 1.

An agent which is able to alter pixels (A_t=true) and situated at position (x_t, y_t) undergoes an attractive force $\overrightarrow{F}$. This force is expressed by the gradient vector of the potential field:

$$\overrightarrow{F} = \begin{cases} -\overrightarrow{\nabla}\Psi(x_t, y_t) \text{ if } A_t\text{=true} \\ \overrightarrow{0} \qquad\qquad \text{otherwise} \end{cases} \tag{4}$$

So, the agent movements, which are stochastic in nature, are weighted by the attractive force applied by the potential field. Agents are influenced to head for the pixels of interest, while keeping random moves. The random component of the agent moves allows the exploration of all regions of the image.

A Relaxation mechanism of potential field is also introduced. It allows the agents gathered around pixels of interest to be released and thus to explore other regions of the image. Around a given pixel, the field intensity decreases after every alteration of this pixel. The equation of the relaxation dynamic is expressed as follows:

$$\Psi_{t+1}(x, y) = \mu \times \Psi_t(x, y), \mu < 1 \tag{5}$$

$\Psi_0(x, y)$ corresponds to the created field after the first alteration of the pixel. The constant μ set to 0.9, represents the decrease rate of the field intensity. After several alignments of a given pixel, the field intensity around this pixel decreases, and tends to zero. This situation represents the final state of the process, after which the system can be stopped.

3.3 Edge Detection and Noise Removal

While moving over the image, an agent smoothes the pixels that approximately belong to its planar region and considers all other pixels as noise pixels. Among these latter, the agent systematically aligns the first encountered one to its current region. However, pixels on the boundaries of regions are true-edge pixels, and thus should not be aligned. Nevertheless, the competition between agents preserves these pixels against an inappropriate smoothing. Indeed, around an edge between two adjacent planar regions, two groups of agents are formed on the two sides of the edge. Each group is formed of agents passing from one region to the other. Agents of each group align the pixels of the edge to their respective region. So, the pixels of the edge are continuously swapped between the two adjacent regions. The resulting alternative alignment of edge pixels allows these pixels to remain emergent in the image. This pattern of competitive actions between agents allows the emergence of the edges in the image, whose detection is not coded in any agent, but results from the collective action of all the agents.

Unlike the true regions of the image, which remain preserved against erasing, the noise regions continuously narrow, and they finally disappear. The borders of these regions are continuously aligned to the true planar regions that surround them. An agent, having aligned a pixel that belongs to the border of a noise region, and having moved inside it, will not be able to adapt. Consequently, the agent cannot align any pixel when leaving the noise region. This occurs in two distinct situations: 1) when the region is planar but insufficiently large to allow agents to cross the minimal path-length L, necessary to be able to adapt; 2) when the region is sufficiently large but not planar, or made up of random depths (noise). In both situations, the agent leaves the noise region and will adapt inside the surrounding planar one. As a summary we can say that true regions have large sizes, sufficient to allow agents to adapt and then align the boundary pixels when leaving these regions. However, noise regions, which are

non planar or having weak sizes, prevent agents from adapting. Consequently, agents will be unable to align pixels on the boundaries of these regions when leaving them. As a result, the borders of a noise region are continuously aligned from outside by including their pixels in the true surrounding regions. After several iterations, all the noise regions will be completely erased. At the end of the process, all the regions in the image are well delimited by the detected boundaries. A simple region growing, steered by the detected boundaries, allows extracting the regions of the image.

4 Experimentation and Analysis

We have used a well known dedicated framework for the evaluation of range image segmentation algorithms [5]. The framework allows to compare a machine-generated segmentation (MS) with a manually-generated segmentation, supposed ideal and representing the ground truth (GT). The most important performance evaluation metrics are the numbers of instances respectively of correctly detected regions, over-segmented regions, under-segmented regions, missed regions, and noise regions. Region classification is performed according to a compare tool tolerance T; $50\% < T \leq 100\%$, which reflects the strictness of the classification. In our case, four methods, namely USF, WSU, UB and UE, cited in [5] are involved in the comparison.

For our method, named 2ARIS, for Agent-based Approach for Range Image Segmentation , six parameters should be set: ξ_{min}, ξ_{max}, Tr_θ, Tr_D, N, and L. These parameters are divided into two subsets: 1) ξ_{min}, ξ_{max}, Tr_θ, and Tr_D representing respectively the two alignment strengths, the angle threshold, and the depth threshold. 2) N and L are respectively the number of agents, and the adaptation path-length. For the first parameter subset, 256 combinations namely $(\xi_{min}, \xi_{max}, Tr_\theta, Tr_D) \in \{0.5, 0.3, 0.1, 0.05\} \times \{1.0, 3.0, 5.0, 7.0\} \times \{15, 18, 21, 24\} \times \{12, 16, 20, 24\}$ were run on the training images. The performance criterion for these parameters is the average number of the correctly detected regions, with the compare tool tolerance T set to 80%. The two alteration strengths ξ_{min} and ξ_{max} are set respectively to 0.3 and 5.0. The thresholds Tr_θ, Tr_D were respectively set to 21, and 16. Note that inappropriate values of N and L can result in a high rate of segmentation errors. Indeed, an insufficient number of agents (N) lead to an under-processing of the image. So, resulting regions are deprived of a set of pixels, which should be included in these regions. A low value of the adaptation path-length L leads to take into account small planar regions, which should be considered as noise regions. However, higher values of L can lead to consider some true planar regions, which are insufficiently large, as noise regions (see section 3.3). In order to set the parameters N and L, 25 combinations of these parameters, namely $(N, L) \in \{1500, 2000, 2500, 3000, 3500\} \times \{3, 5, 7, 9, 11\}$ were run on the training set. In this case, the performance criterion is the average number of noise regions, with the compare tool tolerance set to 80%. Obtained optimal values of N and L are respectively 2500 and 7.

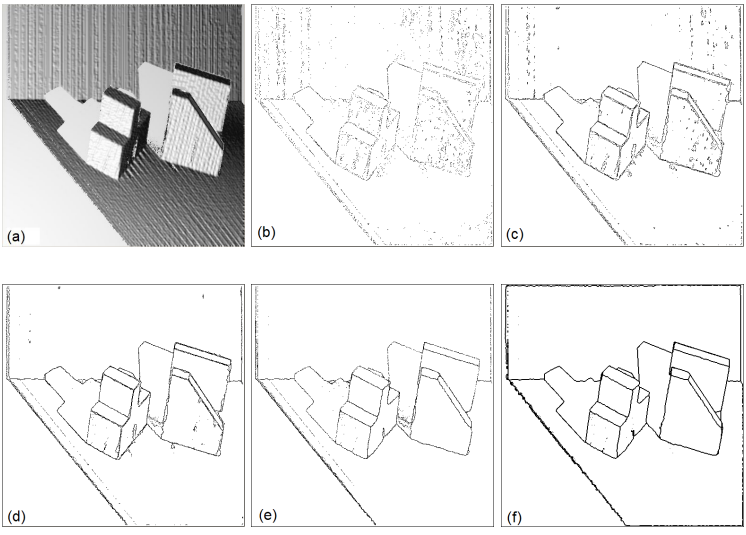


Fig. 2. Segmentation progression. (a) Rendered range image (abw.test.6); (b) at t=1000, (c) at t=5000; (d) at t=9000; (e) at t=13000; (f) Extracted regions.

Table 1. Average results of the different involved methods with T=80%

Method	GT	Correct	Over-seg	Under-seg	Missed	Noise
USF	15.2	12.7	0.2	0.1	2.1	1.2
WSU	15.2	9.7	0.5	0.2	4.5	2.2
UB	15.2	12.8	0.5	0.1	1.7	2.1
UE	15.2	13.4	0.4	0.2	1.1	0.8
2ARIS	15.2	13.0	0.5	0.1	1.4	0.9

Fig. 2 shows an instance of segmentation progression within time. The time t represents the number of steps performed by each agent since the beginning of the process. Displaying a range image by a simple rendering algorithm (Fig. 2a), allows to notice the high level of noise in the used images. Figures 2b, 2c, 2d and 2e show the set of pixels of interest (edge or noise pixels) respectively at t=1000, 5000, 9000 and 13000. Regions are progressively smoothed by aligning the noise pixels to the surrounding planar regions. Edges between adjacent regions are also progressively thinned. At the end of the process, region borders consist of thin lines of one pixel wide (Fig. 2e). Fig. 2f shows the segmentation result obtained by displaying the borders of the extracted regions.

Table 1 contains the average results obtained with all the test images, and for all the performance metrics. The compare tool tolerance was set to the typical value 80%. By considering both the correct detection and the incorrect detection metrics, obtained results show the good efficiency of our method. For all the incorrect detection metrics (instances of Over-segmentation, Under-segmentation, Missed Region, Noise Region), our method has equivalent scores to those of UE

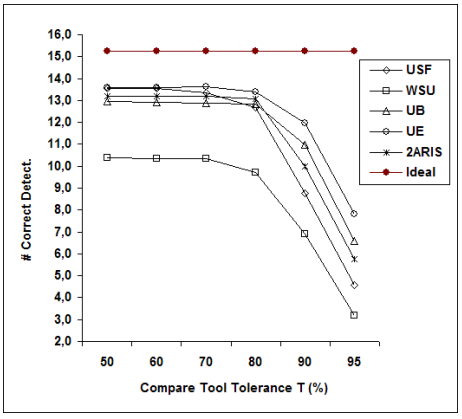


Fig. 3. Average results of correctly detected regions of all methods

and USF. The two latter scored higher than UB and WSU. Fig. 3 shows the average numbers of correctly detected regions for all the test images according to the compare tool tolerance T. Results show that the number of the correctly detected regions by our method is in average better than those of USF, UB and WSU. For instance, our method scored higher than WSU for all the values of the compare tool tolerance T. It scored higher than USF for $T \in \{80\%, 90\%, 95\%\}$, and better than UB for $T \in \{50\%, 60\%, 70\%, 80\%\}$.

5 Conclusion

In this paper, we have introduced a multi-agent approach for range image segmentation. Competitive actions between agents have allowed the emergence of the edges in the image. Image edges, for which no explicit detection was coded in any agent, result from the collective action of all the agents. Obtained results are better than those provided by the traditional algorithms. Moreover, used agents are weakly coupled and indirectly communicate via the environment (image). This allows parallel or distributed implementations, suited for a high computational efficiency. The experimental results obtained with real images from the ABW database show the potential of the proposed method for an efficient and accurate segmentation of range images. The average run time is 8 sec., on a Compaq PC n×8220. The recorded run times were better that those provided by region based methods, and equivalent to edge-based ones. The proposed method should record interesting run times, when implemented on massively parallel architectures, such as Transputer-based architectures. The proposed approach can be extended to deal with more complex surfaces by defining their specific proprieties, and endowing the agents with the appropriate behavior.

References

1. Bab Hadiashar, A., Gheissari, N.: Range image segmentation using surface selection criterion. IEEE Transactions on Image Processing 15(7), 2006–2018 (2006)
2. Bovenkamp, E.G.P., Dijkstra, J., Bosch, J.G., Reiber, J.H.C.: Multi-agent segmentation of IVUS images. Pattern Recognition 37(4), 647–663 (2004)
3. Ding, Y., Ping, X., Hu, M., Wang, D.: Range image segmentation based on randomized hough transform. Pattern Recognition Letters 26(13), 2033–2041 (2005)
4. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
5. Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P.J., Bunke, H., Goldgof, D.B., Bowyer, K.W., Eggert, D.W., Fitzgibbon, A.W., Fisher, R.B.: An experimental comparison of range image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(7), 673–689 (1996)
6. Inokuchi, S., Nita, T., Matsuda, F., Sakurai, Y.: A three dimensional edge-region operator for range pictures. In: 6th International Conference on Pattern Recognition, Munich, pp. 918–920 (1982)
7. Jiang, X., Bunke, H.: Edge detection in range images based on Scan Line approximation. Computer Vision and Image Understanding 73(2), 183–199 (1999)
8. Kang, S.B., Ikeuchi, K.: The complex EGI: A new representation for 3-D pose determination. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(7), 707–721 (1993)
9. Krishnamurthy, E.V., Murthy, V.K.: Distributed agent paradigm for soft and hard computation. Journal of Network and Computer Applications 29(2), 124–146 (2006)
10. Li, S., Zhao, D.: Gradient-based polyhedral segmentation for range images. Pattern Recognition Letters 24(12), 2069–2077 (2003)
11. Liu, J., Tang, Y.Y.: Adaptive image segmentation with distributed behavior-based agents. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(6), 544–551 (1999)
12. Richard, N., Dojat, M., Garbay, C.: Automated segmentation of human brain MR images using a multi-agent approach. Artificial Intelligence in Medicine 30(2), 153–176 (2004)
13. Tsuji, T., Tanaka, Y., Morasso, P., Sanguineti, V., Kaneko, M.: Bio-mimetic trajectory generation of robots via artificial potential field with time base generator. IEEE Transactions on Systems, Man, and Cybernetics, Part C 32(4), 426–439 (2002)

Abstractions of Multi-agent Systems

Constantin Enea[1,2] and Catalin Dima[1]

[1] LACL, University Paris 12, 61 av. du Général de Gaulle, 94010 Créteil, France
[2] University "Al. I. Cuza", Str. General Berthlot 16, 740083, Iaşi, Romania

Abstract. With the recent development of many model-checkers for the temporal logic of knowledge, abstraction techniques are necessary to increase the size of the systems that can be verified. In this paper, we introduce several abstraction techniques for interpreted systems and we prove several preservation results. These results consider the temporal logic of knowledge under Kleene's 3-valued interpretation along infinite and maximal finite paths.

1 Introduction

Abstraction techniques, often based on abstract interpretation [2], provide a method for symbolically analyzing systems using the abstract instead of the concrete domain. Familiar data-flow analysis algorithms are examples of abstract interpretation. In particular, abstract interpretation can be used in order to build the abstract state-space of the system.

With the recent development of many model-checkers for the temporal logic of knowledge [5,8], abstraction techniques are necessary to increase the size of the systems that can be verified. To our knowledge, this is the first attempt to provide abstraction techniques that are useful to prove properties expressed in the temporal logic of knowledge. We include in our logic all the future and past operators [9], paths quantifiers and knowledge operators [4] and we interpret it over infinite paths or maximal finite paths. Moreover, we use Kleene's 3-valued interpretation, whose third truth value $\perp$ is very useful in modeling incomplete observations or malfunctions and performing abstractions. We study three forms of property preservation, the first two being frequently found in the literature [3]: weak-preserving that is preserving the truth value 1, error-preserving that is preserving the truth value 0 and very-weak preserving that is preserving $\{\perp, 1\}$.

The paper is organized as follows. Section 2 fixes the notation and terminology with respect to the temporal logic of knowledge under Kleene's 3-valued interpretation along infinite or maximal finite paths, and 3-valued multi-agent Kripke structures. Section 3 is dedicated to abstractions of multi-agent Kripke structures and the next section deals with preservation results. The paper ends with some concluding remarks.

2 3-Valued KCTL*P

The necessity of using multi-valued logics and multi-valued Kripke structures to model and reason about systems with or without knowledge has been pointed

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 11–21, 2007.

out by many researchers [1,7]. In this paper we use 3-valued multi-agent Kripke structures and a 3-valued temporal logic of knowledge interpreted on infinite paths or maximal finite paths in order to model and reason about systems.

Let AP be a set of atomic propositions. The syntax of $\mathsf{KCTL}^*\mathsf{P}$ is as follows:

$$\phi = \psi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathbf{X}\ \phi \mid \overline{\mathbf{X}}\ \phi \mid \mathbf{G}\ \phi \mid \mathbf{F}\ \phi \mid \phi\ \mathbf{U}\ \phi \mid \phi\ \mathbf{W}\ \phi \mid \mathbf{Pr}\ \phi \mid \\ \mid \mathbf{A}\ \phi \mid \mathbf{O}\ \phi \mid \phi\ \mathbf{S}\ \phi \mid \phi\ \mathbf{B}\ \phi \mid \exists\phi \mid \forall\phi \mid \mathbf{K}_i\phi \mid \mathbf{P}_i\phi,$$

where ψ is a propositional formula over AP and $1 \leq i \leq n$ (n is the number of agents in the system). We use *future time operators* [9]: $\mathbf{X}$ and $\overline{\mathbf{X}}$ are the strong and weak versions of "next-time", $\mathbf{G}$ is "always", $\mathbf{F}$ is "eventually", $\mathbf{U}$ is "until", and $\mathbf{W}$ is "waiting-for"; *past time operators* [9]: $\mathbf{Pr}$ is "previous", $\mathbf{A}$ is "always in the past", $\mathbf{O}$ is "once", $\mathbf{S}$ is "since", and $\mathbf{B}$ is "back to"; *path quantifiers*: $\forall$ and $\exists$; and *knowledge operators* [4]: $\mathbf{K}_i$ means "agent i knows ϕ" and $\mathbf{P}_i$ means "agent i thinks that ϕ is possible". We denote by $\forall\mathsf{KCTL}^*\mathsf{P}$ the subset of $\mathsf{KCTL}^*\mathsf{P}$ consisting of formulas that do not contain $\exists$, $\neg$ and $\mathbf{P}_i$, and by $\exists\mathsf{KCTL}^*\mathsf{P}$ the subset of formulas that do not contain $\forall$, $\neg$, $\mathbf{K}_i$.

The semantics we choose for $\mathsf{KCTL}^*\mathsf{P}$ is based on *Kleene's 3-valued interpretation* [6] and 3-valued multi-agent Kripke structures. Kleene's 3-valued interpretation consists of the set of truth values $\{0, 1, \perp\}$ ordered as $0 < \perp < 1$ and the interpretation of the propositional operators is defined by $x \vee y = max\{x, y\}$ and $x \wedge y = min\{x, y\}$, for any truth values x and y.

Definition 1. *A* 3-valued multi-agent Kripke structure *over a set of atomic propositions AP is a tuple $M = (Q, R, L, (\sim_i \mid 1 \leq i \leq n))$, where Q is the set of states, $R : Q \times Q \to \{0, 1, \perp\}$ is the transition predicate, $L : Q \times AP \to \{0, 1, \perp\}$ is the labeling function which associates to any pair (q, p) the truth value of p in state q, n is the number of agents and $\sim_i : Q \times Q \to \{0, 1, \perp\}$, for any $1 \leq i \leq n$, is the similarity relation of agent i (it gives the degree of similarity between two states from the point of view of agent i). Every relation $\sim_i$ is a* 3-valued equivalence*, i.e. it satisfies* reflexivity*: $\sim_i (x, x) = 1$, for any $x \in Q$;* symmetry*: $\sim_i (x, y) = \sim_i (y, x)$, for any $x, y \in Q$;* transitivity*: $\sim_i (x, z) = \sim_i (x, y) \wedge \sim_i (y, z)$, for any $x, y, z \in Q$.*

$\perp$ values in the definition of a 3-valued multi-agent Kripke structure are useful for two main reasons: *modeling systems with incomplete observations or malfunctions* as mentioned also in [1,7] and *performing abstractions*. The states of an abstract system correspond to sets of states of the original system. The transition between two abstract states depends on the sets of states abstracted by these abstract states. For example, if two disjoint sets of states A and B are abstracted by q_A and q_B, respectively, and for any state in A there exist transitions to all states in B, then we have a definite transition from q_A to q_B; otherwise, we may use a $\perp$-transition from q_A to q_B just to specify that it might be possible to transit from some state in A to some state in B. A similar discussion holds also for the labeling function and the similarity relations.

A *path* of a 3-valued multi-agent Kripke structure is either an infinite sequence of states $\pi = q_0 q_1 \ldots$, such that $R(q_i, q_{i+1}) \in \{\perp, 1\}$, for all $i \geq 0$, or a maximal finite sequence of states $\pi = q_0 q_1 \ldots q_m$, such that $R(q_i, q_{i+1}) \in \{\perp, 1\}$, for all

$0 \leq i < m$, and $R(q_m, q) = 0$, for all $q \in Q$. The *length* of π is ∞, in the first case, and $m+1$, in the second case. $|\pi|$ stands for the length of path π, the ith element of path π is denoted $\pi(i)$, and π^i is the suffix of π starting at q_i, for any $0 \leq i < |\pi|$. We will denote by $\mathsf{Paths}(M)$ the set of paths of the Kripke structure M and by $\mathsf{Paths}(M, q)$ the set of paths of M starting in state q. Also, we will call a *point* any pair (π, m), where $\pi \in \mathsf{Paths}(M)$ and $m \in \mathbb{N}$ with $0 \leq m < |\pi|$. The set of points of the Kripke structure M will be denoted by $\mathsf{Points}(M)$.

The *interpreted system* corresponding to a multi-agent Kripke structure M is a triple $I = (\mathsf{Paths}(M), L^I, (\sim_i^I |\ 1 \leq i \leq n))$, where

$L^I : \mathsf{Points}(M) \times AP \to \{0, 1, \bot\}$ is the interpretation function for the atomic propositions in M defined by $L^I((\pi, m), p) = L(\pi(m), p)$, for any $\pi \in \mathsf{Paths}(M)$ and $0 \leq m < |\pi|$ and $\sim_i^I$: $\mathsf{Points}(M) \times \mathsf{Points}(M) \to \{0, 1, \bot\}$, for any $1 \leq i \leq n$, is the similarity relation defined by $\sim_i^I\ ((\pi, m), (\pi', m')) = \sim_i\ (\pi(m), \pi'(m'))$. ¿From now on we will make no distinction between L and its extension to points L^I. The same holds for the similarity relations.

We now give the semantics for the formulas in $\mathsf{KCTL^*P}$, which for the fragment that includes only future time is similar to the multi-valued semantics from [7]. If ϕ is a formula in $\mathsf{KCTL^*P}$ and $I = (\mathsf{Paths}(M), L, (\sim_i |\ 1 \leq i \leq n))$ an interpreted system we define the *truth value of* ϕ *in the point* (π, m), denoted by $[\phi]^I_{(\pi,m)}$, in the following way (we give only some of the possible cases, the others being similar):

$$
\begin{array}{ll}
[\psi]^I_{(\pi,m)} & = \text{obtained as usual using the interpretation function } L; \\
[\neg\phi]^I_{(\pi,m)} & = \neg[\phi]^I_{(\pi,m)}; \\
[\phi_1 \wedge \phi_2]^I_{(\pi,m)} & = [\phi_1]^I_{(\pi,m)} \wedge [\phi_2]^I_{(\pi,m)}; \\
[\mathbf{X}\ \phi]^I_{(\pi,m)} & = |\pi^m| > 1 \wedge R(\pi(m), \pi(m+1)) \wedge [\phi]^I_{(\pi,m+1)}; \\
[\overline{\mathbf{X}}\ \phi]^I_{(\pi,m)} & = |\pi^m| \leq 1 \vee (R(\pi(m), \pi(m+1)) \wedge [\phi]^I_{(\pi,m+1)}); \\
[\phi_1\ \mathbf{U}\ \phi_2]^I_{(\pi,m)} & = [\phi_2]^I_{(\pi,m)} \vee \bigvee_{0<i<|\pi^m|}([\phi_2]^I_{(\pi,m+i)} \wedge \\
 & \quad \bigwedge_{0\leq j<i}([\phi_1]^I_{(\pi,m+j)} \wedge R(\pi(m+j), \pi(m+j+1)))); \\
[\mathbf{Pr}\ \phi]^I_{(\pi,m)} & = m = 0 \vee (R(\pi(m-1), \pi(m)) \wedge [\phi]^I_{(\pi,m-1)}); \\
[\phi_1\ \mathbf{S}\ \phi_2]^I_{(\pi,m)} & = [\phi_2]^I_{(\pi,m)} \vee \bigvee_{0\leq i<m}([\phi_2]^I_{(\pi,i)} \wedge \\
 & \quad \bigwedge_{i<j\leq m}([\phi_1]^I_{(\pi,j)} \wedge R(\pi(j-1), \pi(j)))); \\
[\exists\phi]^I_{(\pi,m)} & = \bigvee_{\pi'[1..m]=\pi[1..m]}[\phi]^I_{(\pi',m)}; \\
[\forall\phi]^I_{(\pi,m)} & = \bigwedge_{\pi'[1..m]=\pi[1..m]}[\phi]^I_{(\pi',m)}; \\
[\mathbf{K}_i\phi]^I_{(\pi,m)} & = \bigwedge_{\sim_i((\pi,m),(\pi',m'))\neq 0}\big(\sim_i((\pi,m),(\pi',m')) \wedge [\phi]^I_{(\pi',m')}\big); \\
[\mathbf{P}_i\phi]^I_{(\pi,m)} & = \bigvee_{\sim_i((\pi,m),(\pi',m'))\neq 0}\big(\sim_i((\pi,m),(\pi',m')) \wedge [\phi]^I_{(\pi',m')}\big).
\end{array}
$$

Also, we say that a property $\phi \in \mathsf{KCTL^*P}$ is *valid* in some interpreted system I if $[\phi]^I_{(\pi,m)} = 1$, for all $(\pi, m) \in \mathsf{Points}(M)$. If there exists some point (π, m) such that $[\phi]^I_{(\pi,m)} = 0$, we will say that ϕ is *false* in I.

Running Example. Consider the system from Figure 1. It consists of two agents H and L asynchronously composed (the $\wedge$ symbol means that the instructions are executed in one atomic transition). H continuously increases x by

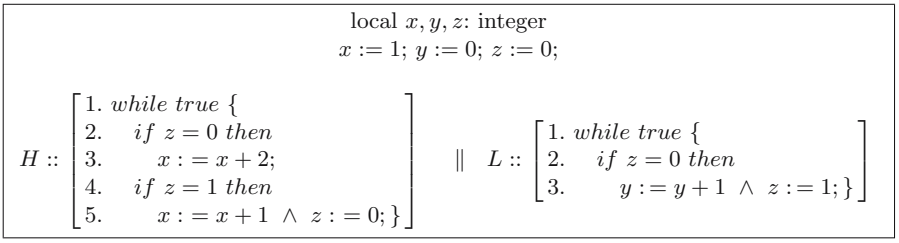


Fig. 1. A system with two processes

2 but, when it receives a signal from L (L sets z to 1) it increases x only by 1. The agent L also uses y to count the number of signals it sends.

Suppose that L can not read x. We would be interested in checking whether the low-level user L can deduce something about the parity of x.

We model this system by a multi-agent Kripke structure $M = (Q, R, L, (\sim_H, \sim_L))$, whose states are triples (x, y, z), where x, y and z are the variables from Figure 1. We consider that $(x, y, z) \sim_H (x', y', z')$ if $x = x'$ and $z = z'$ and $(x, y, z) \sim_L (x', y', z')$ if $y = y'$ and $z = z'$.

An instance of the question above is the formula

$$\phi = P_L\ \mathbf{O}\ \big((even(y) \wedge z = 1) \Rightarrow \mathbf{P}\ \neg(even(x)\ \ \mathbf{S}\ \ z = 1)\big),$$

where *even* is a predicate that it is 1 for even numbers and 0 otherwise. If this property is false then, the fact that y is even implies that x is even in all the states since the last time z was 1. Consequently, if y is even the low-level user knows the parity of x for some of the past states.

Such a property is an instance of information flow w.r.t *run-based secrecy* [10]. In fact, the falsity of ϕ would prove also that agent H does not maintain run-based secrecy with respect to agent L, i.e. there exists an H-local and satisfiable formula ϕ such that $P_L\ \mathbf{O}\ \phi$ is false.

3 Abstractions of Multi-agent Kripke Structures

In this section we will investigate several types of abstractions that can be applied to multi-agent Kripke structures. Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 |\ 1 \le i \le n))$ be a multi-agent Kripke structure over a set AP of atomic propositions. An abstraction of M_1 is based on some equivalence relation ρ on Q_1 that it is used to create *abstract states*. Hence, an abstract state is any equivalence class on Q_1 (we denote $[q]$ the equivalence class of q with respect to ρ). However, this is not sufficient to create an abstract Kripke structure: we still need a definition for the transition predicate, for the labeling function and for the similarity relations. Let $M_2 = (Q_2, R_2, L_2, (\sim_i^2 |\ 1 \le i \le n))$ be a multi-agent Kripke structure with $Q_2 = Q_1/\rho$. Following [11], we say that R_2 is defined in a $\forall\exists$ *manner from* R_1 if

$$R_2([q],[q']) = \begin{cases} 1, \text{ if } (\forall q_1 \in [q])(\forall q_2 \in [q'])(R(q_1, q_2) = 1) \\ 0, \text{ if } (\exists q_1 \in [q])(\exists q_2 \in [q'])(R(q_1, q_2) = 0) \\ \perp, \text{ otherwise,} \end{cases}$$

for any $q, q' \in Q_1$. Defining R_2 in a $\exists\forall$ or $\forall\forall$ manner from R_1 can be done similarly by changing the quantifiers that specify how elements from equivalence classes are chosen. We also consider the three cases for the definition of L_2. For example, L_2 is defined in a $\exists\forall$ *manner from* L_1 if

$$L_2([q], p) = \begin{cases} 1, \text{ if } (\exists q_1 \in [q])(L_1(q_1, p) = 1) \\ 0, \text{ if } (\forall q_1 \in [q])(L_1(q_1, p) = 0) \\ \perp, \text{ otherwise,} \end{cases}$$

for any $q \in Q_1$ and $p \in AP$.

Trying to adopt a similar approach for the similarity relations $\sim_i^2$, we have to ensure that these relations remain three-valued equivalences. The relations between the different possible definitions for $\sim_i^2$ and the properties of a three-valued equivalence are enumerated in the next proposition.

Proposition 1. *Let M_1 be a multi-agent Kripke structure like above, ρ an equivalence relation on Q_1 and $\delta : Q_1/\rho \times Q_1/\rho \to \{0, 1, \perp\}$ a function. We have that:*

1. *if δ is defined in a $\forall\exists$-manner from $\sim_i^1$ then it is symmetric and transitive;*
2. *if δ is defined in a $\exists\forall$-manner from $\sim_i^1$ then it is reflexive and symmetric;*
3. *if δ is defined in a $\forall\forall$-manner from $\sim_i^1$ then it is symmetric;*

The above proposition mentions all the properties of a 3-valued equivalence that are always satisfied by a $\forall\exists$, $\exists\forall$ or $\forall\forall$ definition of $\sim_i^1$. Consequently, when trying to redefine the similarity relations in the abstract system we have to apply some closures to obtain three-valued equivalences.

Definition 2. *Let M_1, ρ and M_2 be as above. M_2 is called a (T_1, T_2, T_3)*-abstraction of M_1 by ρ, *for any $T_1, T_2, T_3 \in \{\forall\forall, \forall\exists, \exists\forall\}$, if R_2 is defined in a T_1 manner from R_1, L_2 is defined in a T_2 manner from L_1 and:*
– *if $T_3 = \forall\exists$, $\sim_i^2$ is the reflexive closure of the $\forall\exists$ definition of $\sim_i^1$;*
– *if $T_3 = \exists\forall$, $\sim_i^2$ is the transitive closure of the $\exists\forall$ definition of $\sim_i^1$;*
– *if $T_3 = \forall\forall$, $\sim_i^2$ is the reflexive and transitive closure of the $\forall\forall$ definition of $\sim_i^1$.*

Running Example. We define on the set of states of M the equivalence relation ρ as follows: $(x, y, z)\ \rho\ (x', y', z')$ if $even(|x - x'|) \wedge even(|y - y'|) \wedge z = z'$. The $(\exists\forall, \exists\forall, \exists\forall)$-abstraction of M by ρ is depicted in Figure 2. We have denoted the abstract states by triples, where the first two elements specify the parity of x and y, respectively, while the third component gives the value of z.

Notice that the abstract system has only 4 reachable states and it is very easy to verify it. We will investigate in the next section the cases in which the truth value of a property is preserved in the concrete system.

4 Preservation Results

This section is dedicated to proving the utility of the types of abstraction presented in the previous section. In general, abstractions are useful when they offer

preservation results with respect to a specific set of properties. We will study three forms of property preserving, the first two being frequently found in the literature [3]:

- *weak-preservation with respect to a set of properties* P: an abstraction is weakly preserving with respect to P if for any $\phi \in P$, ϕ is evaluated to 1 in the abstract system implies that ϕ is evaluated to 1 in the concrete system;
- *error-preservation with respect to a set of properties* P: an abstraction is error preserving with respect to P if for any $\phi \in P$, ϕ is evaluated to 0 in the abstract system implies that ϕ is evaluated to 0 in the concrete system;
- *very weak-preservation with respect to a set of properties* P: an abstraction is very weak-preserving with respect to P if for any $\phi \in P$, ϕ is evaluated to 1 or $\perp$ in the abstract system implies that ϕ is evaluated to 1 or $\perp$ in the concrete system.

4.1 Weak and Very Weak Preservation Results

We will study in this subsection the weak and very weak preservation results that hold for the types of abstraction presented in the previous section. We will give first two technical lemmas that will identify the basic conditions that must be satisfied by the abstraction in order to be weak or very weak preserving with respect to $\forall$KCTL*P or $\exists$KCTL*P.

Let $M_1 = (Q_1, R_1, L_1, (\sim_i^1 \mid 1 \leq i \leq n))$ and $M_2 = (Q_2, R_2, L_2, (\sim_i^2 \mid 1 \leq i \leq n))$ be two 3-valued multi-agent Kripke structures over some set AP of atomic propositions and ρ an equivalence relation on Q_1 such that $Q_2 = Q_1/\rho$. We say that M_2 *preserves the paths of* M_1 if for each path π_1 in M_1 there exists a path π_2 in M_2 with the same length as π_1 such that $\pi_2(m) = [\pi_1(m)]$, for any $0 \leq m < |\pi_2|$ (we say that π_2 *corresponds* to π_1). Also, we say that M_1 *preserves the paths of* M_2 if for each path π_2 in M_2 there exists a path π_1 in M_1 with the same length such that $\pi_1(m) \in \pi_2(m)$, for any $0 \leq m < |\pi_1|$ (again, we say that π_1 *corresponds* to π_2). We say that R_2 (R_1) *preserves the set of truth values* B if $R_2([q],[q']) \in B$ ($R_1(q,q') \in B$) implies $R_1(q,q') \in B$ ($R_2([q],[q']) \in B$), for any $q, q' \in Q_1$. In the same way we define the facts that L_2, L_1, $\sim_i^2$ and $\sim_i^1$ preserve some set of truth values.

Lemma 1. *Let M_1 and M_2 be as above. For any formula ϕ in* $\forall$KCTL*P*:*

1. *if M_2 preserves the paths of M_1 and L_2, $\sim_i^1$ preserve $\{\perp, 1\}$, then $[\phi]^{I_2}_{(\pi_2,m)} \in \{\perp, 1\}$ implies $[\phi]^{I_1}_{(\pi_1,m)} \in \{\perp, 1\}$, for any $(\pi_1, m) \in$ Points(M_1) (π_2 is the*

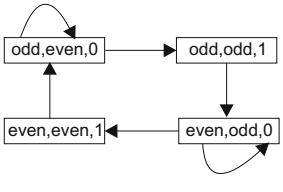


Fig. 2. An $(\exists\forall, \exists\forall, \exists\forall)$-abstraction of M

path in M_2 that corresponds to π_1 and I_1, I_2 are the interpreted systems corresponding to M_1, M_2).

2. *if M_2 preserves the paths of M_1, $\sim_i^1$ preserves $\{\bot, 1\}$ and R_2, L_2, $\sim_i^2$ preserve 1, then $[\phi]^{I_2}_{(\pi_2,m)} = 1$ implies $[\phi]^{M_1}_{(\pi_1,m)} = 1$, for any $(\pi_1, m) \in \mathsf{Points}(M_1)$. The same holds even if the requirements that R_2 and $\sim_i^2$ preserve 1 are replaced by $R_1(q, q') \in \{0, 1\}$ and $\sim_i^1$ $(q, q') \in \{0, 1\}$, for any $q, q' \in Q_1$.*

Proof. All claims can be proved by structural induction on ϕ. We will exhibit only the interesting cases of the first proof, the others being obtained in a similar manner:

- $\phi = p \in AP$. Then, $[\phi]^{I_2}_{(\pi_2,m)} = L_2(\pi_2(m), p) \in \{\bot, 1\}$. By the definition of π_2 and the fact that L_2 preserves $\{\bot, 1\}$, we obtain that $L_1(\pi_1(m), p) \in \{\bot, 1\}$. Therefore, $[\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;
- $\phi = \overline{\mathbf{X}}\ \phi_1$. Assume that ϕ_1 satisfies the property. Then, $[\overline{\mathbf{X}}\ \phi_1]^{I_2}_{(\pi_2,m)} = |\pi_2^m| \leq 1 \vee \left([\phi_1]^{I_2}_{(\pi_2,m+1)} \wedge R_2(\pi_2(m), \pi_2(m+1))\right) \in \{\bot, 1\}$. If $|\pi_2^m| \leq 1$, then $|\pi_1^m| \leq 1$ which leads to $[\overline{\mathbf{X}}\ \phi_1]^{I_1}_{(\pi_1,m)} = 1$. Otherwise, $[\phi_1]^{I_2}_{(\pi_2,m+1)} \wedge R_2(\pi_2(m), \pi_2(m+1)) \in \{\bot, 1\}$. By the definition of π_2 we obtain $|\pi_1^m| > 1$ and $R_1(\pi_1(m), \pi_1(m+1)) \in \{\bot, 1\}$, and by the induction hypothesis, $[\phi_1]^{I_1}_{(\pi_1,m+1)} \in \{\bot, 1\}$. Consequently, $[\overline{\mathbf{X}}\ \phi_1]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$;
- $\phi = \forall\phi_1$. Assume that ϕ_1 satisfies the property. Consequently, $[\forall\phi_1]^{I_2}_{(\pi_2,m)} = \bigwedge_{\pi[1..m]=\pi_2[1..m]} [\phi_1]^{I_2}_{(\pi,m)} \in \{\bot, 1\}$ which implies $[\phi_1]^{I_2}_{(\pi,m)} \in \{\bot, 1\}$, for all paths π such that $\pi[1..m] = \pi_2[1..m]$. Since M_2 preserves the paths of M_1, for each path σ of M_1 with $\sigma[1..m] = \pi_1[1..m]$ there exists a path σ' of M_2 such that $\sigma'[1..m] = \pi_2[1..m]$. Moreover, $[\phi_1]^{I_2}_{(\sigma',m)} \in \{\bot, 1\}$, for each path σ' as above. Applying the induction hypothesis we get $[\phi_1]^{I_1}_{(\sigma,m)} \in \{\bot, 1\}$, for each path σ with $\sigma[1..m] = \pi_1[1..m]$. Hence, $[\forall\phi_1]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$.
- $\phi = K_i\phi_1$. Assume that ϕ_1 satisfies the property. The hypothesis will imply that $\bigwedge_{\sim_i^2((\pi_2,m),(\pi_2'',m'))\neq 0} \sim_i^2 ((\pi_2, m), (\pi_2'', m')) \wedge [\phi]^{I_2}_{(\pi_2'',m')} \in \{\bot, 1\}$, and we have to prove $\bigwedge_{\sim_i^1((\pi_1,m),(\pi_1',m''))\neq 0} \sim_i^1 ((\pi_1, m), (\pi_1', m'')) \wedge [\phi]^{I_1}_{(\pi_1',m'')} \in \{\bot, 1\}$. Let (π_1', m'') be a point of M_1 such that $\sim_i^1 ((\pi_1, m), (\pi_1', m'')) \neq 0$. Since M_2 preserves the paths of M_1, there exists a corresponding path for π_1', denote it π_2' such that $\pi_2'(m'') = [\pi_1'(m'')]$. Moreover, because $\sim_i^1$ preserves $\{\bot, 1\}$ we obtain that $\sim_i^2 ((\pi_2, m), (\pi_2', m'')) \in \{\bot, 1\}$. From above we obtain that $[\phi_1]^{I_2}_{(\pi_2',m'')} \in \{\bot, 1\}$ which, by the induction hypothesis, implies $[\phi_1]^{I_1}_{(\pi_1',m'')} \in \{\bot, 1\}$. □

The following lemma can be proved in a similar way to Lemma 1.

Lemma 2. *Let M_1 and M_2 be as above. For any formula ϕ in $\exists\mathsf{KCTL}^*\mathsf{P}$:*

1. *if M_1 preserves the paths of M_2 and L_2, $\sim_i^2$ preserve $\{\bot, 1\}$, then $[\phi]^{I_2}_{(\pi_2,m)} \in \{\bot, 1\}$ implies $[\phi]^{M_1}_{(\pi_1,m)} \in \{\bot, 1\}$, for any $(\pi_2, m) \in \mathsf{Points}(M_2)$ (π_1 is the path in M_1 that corresponds to π_2).*

2. *if* M_1 *preserves the paths of* M_2 *and* R_2, L_2, $\sim_i^2$ *preserve 1, then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{M_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_2, m) \in \mathsf{Points}(M_2)$.

We will enumerate now the weak and very weak preservation results that hold for all the possible types of abstraction.

Theorem 1. *Let* M_1 *and* M_2 *be as above, where* R_2 *and* $\sim_i^2$, *for any* $1 \leq i \leq n$, *are* $\exists\forall$ *defined. Then, for any* $\phi \in \forall\mathsf{KCTL}^*\mathsf{P}$ *we have the following:*

1. *If* L_2 *is* $\forall\exists$ *defined, then* $[\phi]^{I_2}_{(\pi_2,m)} \in \{\bot, 1\}$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$, *for any* $(\pi_1, m) \in \mathsf{Points}(M_1)$ *(*π_2 *is a path in* M_2 *corresponding to* π_1*). Moreover, if* $R_1(q, q') \in \{0, 1\}$ *and* $\sim_i^1 (q, q') \in \{0, 1\}$, *for all* $q, q' \in Q_1$ *and* $1 \leq i \leq n$, *then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_1, m) \in \mathsf{Points}(M_1)$.
2. *If* L_2 *is* $\forall\forall$ *defined,* $R_1(q, q') \in \{0, 1\}$ *and* $\sim_i^1 (q, q') \in \{0, 1\}$, *for all* $q, q' \in Q_1$ *and* $1 \leq i \leq n$, *then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_1, m) \in \mathsf{Points}(M_1)$.

Proof. Directly from Lemma 1. □

Next, we try to give a weak and a very weak preservation result for abstractions of type $(\forall\exists, T, \forall\exists)$, for some T, with respect to $\exists\mathsf{KCTL}^*\mathsf{P}$. By Lemma 2 we need that $\sim_i^2$ preserves $\{\bot, 1\}$ and since we use the reflexive closure of the $\forall\exists$ definition of $\sim_i^1$, this may not happen. Consequently, the preservation results will hold only for equivalences relations ρ that ensure that the $\forall\exists$ definition of $\sim_i^1$ is already reflexive. If M_1 and M_2 are as above, we say that ρ *is compatible with* $\sim_i^1$ if whenever $q\ \rho\ q'$ we have $\sim_i^1 (q, q') = 1$. Clearly, this compatibility implies that the $\forall\exists$ definition of $\sim_i^1$ is reflexive.

Theorem 2. *Let* M_1 *and* M_2 *be as above, where* R_2 *and* $\sim_i^2$ *are* $\forall\exists$ *defined and* ρ *is compatible with* $\sim_i^1$, *for any* $1 \leq i \leq n$. *Then, for any* $\phi \in \exists\mathsf{KCTL}^*\mathsf{P}$ *we have the following:*

1. *If* L_2 *is* $\forall\exists$ *or* $\forall\forall$ *defined, then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_2, m) \in \mathsf{Points}(M_2)$ *(*π_1 *is a path in* M_1 *corresponding to* π_2*).*
2. *If* L_2 *is* $\forall\exists$ *defined, then* $[\phi]^{I_2}_{(\pi_2,m)} = \bot$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} \in \{\bot, 1\}$, *for any* $(\pi_2, m) \in \mathsf{Points}(M_2)$.

Proof. Directly from Lemma 2. □

Theorem 3. *Let* M_1 *and* M_2 *be as above, where* R_2 *and* $\sim_i^2$, *for any* $1 \leq i \leq n$, *are* $\forall\forall$ *defined. Then, for any formula* ϕ *in* $\forall\mathsf{KCTL}^*\mathsf{P}$ *we have the following:*

1. *If* L_2 *is* $\forall\exists$ *defined, then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_1, m) \in \mathsf{Points}(M_1)$ *(*π_2 *is a path in* M_2 *corresponding to* π_1*).*
2. *If* L_2 *is* $\forall\forall$ *defined, then* $[\phi]^{I_2}_{(\pi_2,m)} = 1$ *implies* $[\phi]^{I_1}_{(\pi_1,m)} = 1$, *for any* $(\pi_1, m) \in \mathsf{Points}(M_1)$.

Proof. Directly from Lemma 1. □

4.2 Error Preservation Results

We will study in this section the error preservation results that hold for the types of abstraction we have introduced in Section 3. As before, we first give two technical lemmas that identify basic conditions that must be satisfied by an abstraction in order to be error-preserving.

Lemma 3. *Let M_1 and M_2 be as above. If M_2 preserves the paths of M_1, L_2 preserves 0 and $\sim_i^1$ preserves $\{\bot, 1\}$, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{M_1}_{(\pi_1,m)} = 0$, for any $\phi \in \exists\mathsf{KCTL}^*\mathsf{P}$ and $(\pi_1, m) \in \mathsf{Points}(M_1)$ (π_2 is the path in M_2 that corresponds to π_1).*

Proof. The claim is proved by structural induction on ϕ. We will enumerate only the interesting cases, the others can be obtained similarly:

- $\phi = \exists\phi_1$. Assume that ϕ_1 satisfies the property. We obtain that $[\exists\phi_1]^{I_2}_{(\pi_2,m)} = \bigvee_{\pi[1..m]=\pi_2[1..m]}[\phi_1]^{I_2}_{(\pi,m)} = 0$ implies that $[\phi_1]^{I_2}_{(\pi,m)} = 0$, for all paths π such that $\pi[1..m] = \pi_2[1..m]$. Since M_2 preserves the paths of M_1, for each path σ of M_1 with $\sigma[1..m] = \pi_1[1..m]$ there exists a path σ' of M_2 such that $\sigma'[1..m] = \pi_2[1..m]$. Moreover, $[\phi_1]^{I_2}_{(\sigma',m)} = 0$, for each path σ' as above. Applying the induction hypothesis we get $[\phi_1]^{I_1}_{(\sigma,m)} = 0$, for each path σ with $\sigma[1..m] = \pi_1[1..m]$. Hence, $[\exists\phi_1]^{I_1}_{(\pi_1,m)} = 0$
- $\phi = P_i\phi_1$. Assume that ϕ_1 satisfies the property. The hypothesis will imply that $\bigvee_{\sim_i^2((\pi_2,m),(\pi_2'',m'))\neq 0}\big(\sim_i^2((\pi_2,m),(\pi_2'',m')) \wedge [\phi]^{I_2}_{(\pi_2'',m')}\big) = 0$ and we have to prove $\bigvee_{\sim_i^1((\pi_1,m),(\pi_1',m''))\neq 0} \sim_i^1((\pi_1,m),(\pi_1',m'')) \wedge [\phi]^{I_1}_{(\pi_1',m'')} = 0$. Let (π_1', m'') be a point of M_1 such that $\sim_i^1((\pi_1,m),(\pi_1',m'')) \neq 0$. Since M_2 preserves the paths of M_1, there exists a corresponding path for π_1', denote it π_2' such that $\pi_2'(m'') = [\pi_1'(m'')]$. By the fact that $\sim_i^1$ preserves $\{\bot, 1\}$, we obtain $\sim_i^2((\pi_2,m),(\pi_2',m'')) \in \{\bot, 1\}$. ¿From above, $[\phi_1]^{I_2}_{(\pi_2',m'')} = 0$ and by the induction hypothesis we get $[\phi_1]^{I_1}_{(\pi_1',m'')} = 0$. □

The following lemma can be proved in a similar way to Lemma 3.

Lemma 4. *Let M_1 and M_2 be as above. If M_1 preserves the paths of M_2, L_2 preserves 0 and $\sim_i^2$ preserves $\{\bot, 1\}$, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{M_1}_{(\pi_1,m)} = 0$, for any $\phi \in \forall\mathsf{KCTL}^*\mathsf{P}$ and $(\pi_2, m) \in \mathsf{Points}(M_2)$ ($\pi_1 \in \mathsf{Paths}(M_1)$ corresponds to π_2).*

We can now state the error preservation results.

Theorem 4. *Let M_1 and M_2 be as above, where R_2 and $\sim_i^2$ are $\exists\forall$ defined. If L_2 is $\exists\forall$ or $\forall\forall$ defined, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{I_1}_{(\pi_1,m)} = 0$, for any $\phi \in \exists\mathsf{KCTL}^*\mathsf{P}$ and $(\pi_1, m) \in \mathsf{Points}(M_1)$ ($\pi_2 \in \mathsf{Paths}(M_2)$ corresponds to π_1).*

Proof. Directly from Lemma 3. □

Next, we prove that the $(\forall\exists, T, \forall\exists)$-abstractions, for some T, are error preserving with respect to $\forall\mathsf{KCTL}^*\mathsf{P}$. Again, ρ must be compatible with $\sim_i^1$.

Theorem 5. *Let M_1 and M_2 be as above, where R_2 and $\sim_i^2$ are $\forall\exists$ defined and ρ is compatible with $\sim_i^1$, for any $1 \leq i \leq n$. If L_2 is $\exists\forall$ or $\forall\forall$ defined, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{I_1}_{(\pi_1,m)} = 0$, for any $\phi \in \forall\mathsf{KCTL}^*\mathsf{P}$ and $(\pi_2, m) \in \mathsf{Points}(M_2)$ (π_1 is the path in M_1 corresponding to π_2).*

Proof. Directly from Lemma 4. □

Theorem 6. *Let M_1 and M_2 be as above, where R_2 and $\sim_i^2$, for any $1 \leq i \leq n$, are $\forall\forall$ defined. For any ϕ in $\exists\mathsf{KCTL}^*\mathsf{P}$ we have that:*

1. *If L_2 is $\exists\forall$ defined, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{I_1}_{(\pi_1,m)} = 0$, for any $(\pi_1, m) \in \mathsf{Points}(M_1)$ (π_2 is a path in M_2 that corresponds to π_1).*
2. *If L_2 is $\forall\forall$ defined, then $[\phi]^{I_2}_{(\pi_2,m)} = 0$ implies $[\phi]^{I_1}_{(\pi_1,m)} = 0$, for any $(\pi_1, m) \in \mathsf{Points}(M_1)$.*

Proof. Directly from Lemma 3. □

Running Example. Using the predicate $odd(x)$, which is 1 for odd numbers and 0 otherwise, we transform ϕ into an equivalent formula which is in $\exists\mathsf{KCTL}^*\mathsf{P}$. The abstract system from Figure 2 has only 4 reachable states and we can easily prove that the formula ϕ is false. Since $\phi \in \exists\mathsf{KCTL}^*\mathsf{P}$ we apply the error preservation result from Theorem 4 and obtain that it is false also in the concrete system I_1.

5 Conclusions

We have introduced in this paper abstraction techniques that are useful to prove properties expressed in the temporal logic of knowledge. We have used a very general logic under Kleene's 3-valued interpretation and we have offered weak, very weak and error preservation results. These techniques can be used to increase the size of the systems that can be verified.

References

1. Chechik, M., Easterbrook, S.M., Petrovykh, V.: Model-checking over multi-valued logics. In: Oliveira, J.N., Zave, P. (eds.) FME 2001. LNCS, vol. 2021, pp. 72–98. Springer, Heidelberg (2001)
2. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of POPL, pp. 238–252 (1977)
3. Dams, D.: Abstract Interpretation and Partial Refinement for Model Checking. PhD thesis, Technische Universität Eindhoven (1996)
4. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
5. Gammie, P., van der Meyden, R.: MCK: Model checking the logic of knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)

6. Ginsberg, M.: Multivalued logics. A uniform approach to inference in artificial intelligence. Computational Intelligence 4, 265–316 (1988)
7. Konikowska, B., Penczek, W.: Model checking for multivalued logic of knowledge and time. In: Proceedings of AAMAS, pp. 169–176 (2006)
8. Lomuscio, A., Raimondi, F.: MCMAS: A model checker for multi-agent systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006 and ETAPS 2006. LNCS, vol. 3920, pp. 450–454. Springer, Heidelberg (2006)
9. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Specification. Springer, Heidelberg (1992)
10. O'Neill, K.R., Halpern, J.Y.: Secrecy in multiagent systems. CoRR, cs.CR/0307057 (2003)
11. Tiplea, F.L., Enea, C.: Abstractions of data types. Acta Informatica 42(8-9), 639–671 (2006)

Agent-Based Network Protection Against Malicious Code

Martin Rehák[1], Michal Pěchouček[2], Jan Tožička[2], Magda Prokopová[1], David Medvigy[2], and Jiří Novotný[3]

[1] Center for Applied Cybernetics, Faculty of Electrical Engineering
[2] Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague Technická 2, 166 27 Prague, Czech Republic
{mrehak,pechouc,prokopova,tozicka}@labe.felk.cvut.cz
[3] Institute of Computer Science, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
novotny@ics.muni.cz

Abstract. This paper presents an agent-based approach to Network Intrusion Prevention on corporate networks, emphasizing the protection from fast-spreading mobile malicious code outbreaks (e.g. worms) and related threats. Agents are not only used as a system-integration platform, but we use modern agent approaches to trust modeling and distributed task allocation to efficiently detect and also counter the attack by automatically created and deployed filters. The ability of the system to react autonomously, without direct human supervision, is crucial in countering the fast-spreading worms, that employ efficient scanning strategies to immediately spread farther once they infect a single host in the network.[1]

1 Introduction

This paper presents an application of classic agent technology techniques: trust modeling, meta-agents, computational reflection and distributed task allocation to protect a local or campus network against the spread of malicious code, e.g. worms [1]. The worms are a specific type of malicious code, that spreads across the computer networks, and uses known vulnerabilities of widely deployed software to infect the hosts, possibly perform malicious actions and spread further. The spread of the worm is influenced by its scanning strategy. Scanning strategy uses the current IP address as an input, and determines the addresses to propagate to, including the order in which the propagation is attempted. Strategies are typically defined in terms of various types of subnets [2] – for example, a Slammer worm uses a random scanning over the whole IPv4 address space, while the Code Red II scans local subnet with higher probability.

[1] This material is based upon work supported by the European Research Office of the US Army under Contract No. N62558-07-C-0001 and N62558-07-C-0007. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Research Office of the US Army. Also supported by Czech Ministry of Education grants 1M0567 and 6840770038.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 22–31, 2007.

Historically, there were many attempts to use agents in intrusion detection systems (IDS) [3]. However, these attempts have never matured into widely-deployed commercial solutions, and we argue that the computational power and techniques were not advanced enough to succeed. In our work, we aim to present a system design that efficiently protects the small and medium-sized networks against the worm attacks with minimum administrative overhead. Therefore, the performance and autonomy of the solution are crucial from the operational point of view.

Our detection approach, presented in Section 3 exploits the weak spot of worm propagation, the randomness of scanning strategy that determines the IP addresses targeted by the worm for future infection attempts. The randomness makes the spread less efficient, due to the exploration of non-existing hosts, and generates significant network traffic that can be detected and matched with the alarms raised by protected hosts (see Section 2). The matching is performed by *IDS Agents*, that use their *extended trust models* [4] to assess the trustfulness/maliciousness of individual network flows. The conclusions of the models are used by IDS agents to generate filtering policies, and to convert these policies into the bytecode (JAVA) by means of computational reflection. Filters are then deployed on network devices to limit the future spread of the worm by means of collective reflection process, based on established distributed task allocation methods [5].

2 System Architecture and Observation

The system, as presented in Fig 1, is designed to fulfill three principal functionalities: to **observe** the network traffic and host behavior, to detect malicious traffic and to react by deploying efficient filters on network devices to prevent the spread of the threat. This functionality is implemented by following elements:

• **Observation** is realized by two types of sensors: - **Host sensors** are deployed on selected *protected hosts* in the network and raise alarms when an exceptional behavior suggests a possibility of attack or intrusion attempt. **Network sensors** are located on network devices and capture the information about the network flows, together with the traffic statistics. The aggregated observations are then processed by IDS agents.

• **Detection** of malicious traffic is the primary responsibility of **IDS agents**, who process the information from the sensors using their trust model [4] in order to correctly correlate the alarms received from hosts with the current network flows. Their

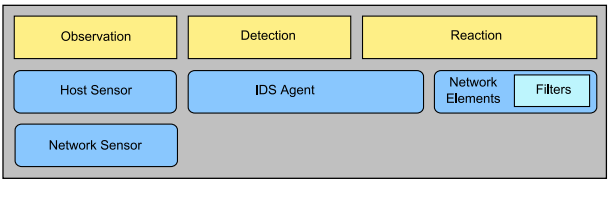


Fig. 1. System Components Overview

role is to correctly identify the malicious traffic and to generate a description of the malicious flow(s) for filter creation in reaction phase.

• **Reaction** phase is triggered upon attack detection by IDS agents. The agents generate one or more filters matching the attack and pass these filters to the network devices. These reflective [6] devices are able to autonomously adapt to changing environment. In our case, the adaptation is ensured by insertion of generated filters into their code and associated delegated operations.

When deployed, the system components use classic agent methods (e.g. directories or matchmakers) to discover each other and to maintain the system operational in the dynamic environment. The essential communication protocol in the observation and detection phase is `SUBSCRIBE` [2], which allows efficient information gathering by the IDS Agents.

The observation of attacks by the Host sensors is based on an assumption of randomized worm spread. As all currently known worms exploit a vulnerability specific for a single system type, we base our defence on an assumption that the deployed hosts are composed of wide variety of system configurations, and that most of these systems ar able to detect an unsuccessful (or even successful) attack, aimed at the vulnerability of other system. The detection of possible attack triggers an alarm that is sent to the IDS agents. We currently assume only simple, binary alarms without any additional information regarding the attack description. This maximizes the range of software/devices able to provide the alarm, and also makes the misuse of the mechanism more difficult, as the potential attacker can not directly specify the ”suspicious” traffic.

At the same time, network sensors capture the information specified in Table 1 about each individual network flow (unidirectional connection component identified by srcIP, dstIP, srcPrt, dstPrt and Protocol) and also provide this information to the same IDS agents that perform the detection process.

Table 1. Characteristics of the flow: Identity and Context (based on [7])

Feature	**Description**	**Feature**	**Description**
Connection Identity		***Connection Context***	
srcIP	Source IP Address.	count-src	Number of flows from the unique sources toward the same destination.
destIP	Destination IP	count-dest	Number of flows to unique destinations from the same source.
srcPort	Source Port	count-serv-dest	Number of flows to the same destination IP using the same source port.
destPort	Destination Port	count-serv-src	Number of flows from the same IP to the same port.
Payload Signature	First 256 bytes of the flow content (application headers)		

[2] www.fipa.org

3 Attack Detection

IDS agents use an extended *trust model* [4,8] to correlate the alarm received from the hosts with the observed traffic and to identify the malicious flows. The trust model inside each IDS agent processes the inputs that are provided by both host and network sensors in near-real time: sensors: (*i*) *network flows* information that contains the TCP/IP headers and first 256 bytes of the flow content [9], (*ii*) *statistics* of the existing network flows, listed in the context part of Table 1, and (*iii*) the *alarms* raised by Host sensors.

The trust model correlates the occurrence of alarms received from the hosts with the features of the relevant captured flows and the statistics associated with a particular flow. The features and the statistics form an *Identity-Context* feature space [4]. As the number of flows in an average network can be significant, it is important to process, represent, and maintain the data efficiently, and to keep the model synchronized with the most recent alerts and traffic. Therefore, instead of associating the information with individual flows in the *Identity-Context* space, the flows are aggregated into clusters. For each cluster, the IDS maintain its trustfulness, expressed as a fuzzy number [10]. The learning of trustfulness in our model is iterative, and each iteration consists of two stages. In the first stage, IDS agents generate and update the clusters using the Leader-Follower[3] algorithm [11], while in the second stage IDS agents update the trustworthiness that is associated with the centroids of the clusters that are adjacent to the observation representation in the Identity-Context space. For sake of efficiency, our implementation actually performs both stages in the same time, as we can see in Alg. 1.

Algorithm 1. Processing of the new observation.

```
Input: flow,situat,trustObs
closest ← nil;
mindist ← ∞;
id ← identityCx(flow,situat)
foreach rc ∈ rclist do
    dist ← distance(rc,id)
    if dist < mindist then
        closest ← rc
    wei = weight(dist)
    if wei > threshold then
        rc.updateTrust(trustObs,wei)
end
if mindist > clustsize then
    rclist.append(id)
    id.updateTrust(trustObs,wei)
else
    closest.updatePosition(id)
```

When IDS agent **observes** a flow, it extracts the identity features (see Tab. 1), then retrieves the associated statistics to determine the position of this vector in the

[3] Our current implementation uses LF clustering as it is efficient in terms of computational and memory cost, and allows on-line processing. Any other clustering algorithm can also be used instead, the reference points can even be placed arbitrarily [8].

Identity-Context space (*id* in Alg. 1) . Then, it computes the distance of the observed flow vector to each of the existing centroids ($rc \in rclist$), and updates the trustfulness of these centroids with a weight *wei* that decreases with growing distance. If the closest centroid is farther away than a predetermined threshold *clustsize*, a new cluster is defined around the new observation. When the model is **queried** regarding the trustfulness of a specific flow vector, it aggregates the trustfulness associated with the centroids, with a weight that decreases with the distance from the centroid to the flow representation in the Identity-Context space. Once it has determined the trustfulness, in a form of a quasi-triangular fuzzy number, it calculates inferences with a high trust and low trust fuzzy intervals. These intervals represent the knowledge about the normal level of trustfulness in the environment, and allow the IDS agent to infer whether the flow is trusted or not under current conditions [10].

The output of the detection phase is a selection of malicious traffic, specified as a cluster (or a group of clusters) in the identity-context space. This description, together with the examples of typical cluster members is then used to generate a traffic filter that only use the identity part of the features from Table 1, which can be matched with actual flows.

4 Reaction to Attack

As the detection uses the actual infections to detect the malicious traffic, we must assume that at least some of the vulnerable hosts in the network were already infected by the detected threat. Therefore, the filter generated by the detection upon attack discovery must not only protect the network interfaces (firewalls, VPN access,...), but shall be deployed at least once on each path between any two hosts on the network. If the filter can not be deployed in some network parts, these parts shall be considered as a single host by the protection mechanism and treated as such.

As there may be multiple filters deployed on the network in the same time, and a big part of network devices may be unable to perform the complete filtering of the passing traffic on their own, we use the distributed task allocation methods to find optimal delegation of filtering tasks between the network devices.

The delegation of filtering is based on negotiation among two or more nodes. As a result of the negotiation one node would be dedicated to provide filtering services to the other nodes. Such extension can increase the number of deployed filters, but increases also the communication traffic (as the flows must be tunneled to filtering devices). In case of delegation, outlined in Extended Filter Deployment Algorithm (EFDA, see Alg. 2), the agent uses Extended Contract Net Protocol (ECNP) [5] to ask trusted agents (determined by policy or a trust model) to perform filtering on its behalf, instead of local filter deployment.

One of the main features of Extended Contract Net Protocol compared to simple Contract Net Protocol [12] is that the participants can propose to fulfill only part of the task. The initiator agent starts negotiation by sending task specification to all trusted agents and is waiting for proposals with specified costs. In case participant agent decides to propose for given task it reserves all resources necessary for fulfilling task or part of it and sends proposal to the initiator. At this point participant agent can't cancel its

Algorithm 2. EFDA Algorithm

$\Theta =$ all trusted agents.
$\Phi =$ set of my filters.
$\Psi = \emptyset$.
repeat
  Send Call-For-Proposals to all agents Θ.
  Wait for all proposals Π.
  Choose the winner A:
    the agent that proposes minimal average price for deploying filters F.
  $\Psi = \Psi \cup \{A\}$
  $\Phi = \Phi \setminus F$
  Send Temporary Accept to agent A.
  Send Reject to agents $\Theta \setminus \{A\}$.
until $\Phi = \emptyset$ *or* $\Pi = \emptyset$;
Send Accept to all agents Ψ.

offer by itself and is waiting for the initiator answer. After initiator receives proposals covering parts of the task, the agent with best proposal is given temporary accept. After temporary accept is received, the agent is committed to the task and is waiting for final accept to start performing the task. Initiator sends rejects to all agents but the accepted one and thus they can cancel reservation of their resources.

New negotiation turn is taken for the remaining part of the task and again all trusted agents receive the task specification. After whole task being distributed among participant agents or getting to the point where no more agents can propose to fulfill any part of the task, the initiator agent accepts a consistent subset of proposals that were previously temporarily accepted. At this point both initiator and participant agents are fully committed.

In our domain the task is represented by all of the filters agent is willing to delegate. Participant agents after receiving the task specification are willing to cooperate and create proposal for as many filters as possible. The cost for each filter combines the distance of the agents (initiator and participant - this distance corresponds to the growth of the traffic in the network) and computational costs/price for deploying the filter. When task contains filter that has already been deployed on participant node the price for deploying is zero and thus the proposed filter cost is based only on distance. According to the strategy we want to follow the importance of distance or computational cost in the final price for the task can be changed. The higher the importance of distance is the less filters can be accommodated in the network, but on the other hand the growth of additional network traffic is less significant.

Our task is to deploy as many filters in the network as possible, but at the same time we are trying to keep the additional network traffic as low as possible. Therefore the delegation of filters starts at the point when no more filters can be deployed on local network device.

Using Filter Delegation. Figure 2 shows simple situation of filter delegation. $Switch\ A$ didn't have enough free computational capacity to deploy all filters available, so the EFDA assigned one of additional filters to $Switch\ B$ located slightly off the way. Therefore every-time traffic is sent from $Host\ 1$ to $Host\ 2$ it is forwarded by $Switch\ A$

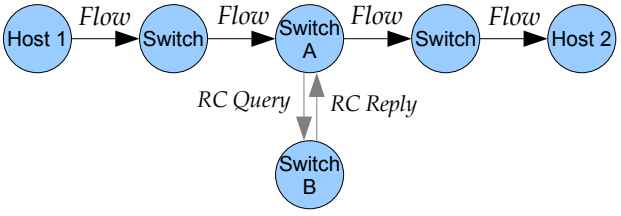


Fig. 2. Using filter delegation $RC\,Query$ is created for traffic from $Host\,1$ to $Host\,2$. $RC\,Query$ contains original flow, which is either returned if determined as not malicious or dropped in other case.

to $Switch\,B$ that contains remaining filters. Special filter called *filter-for* deployed on $Switch\,A$ encloses the traffic in new flow and marked as $RC\,Query$ to be easily recognizable by switch it is addressed to. After receiving $RC\,Query$ device performs filtering and in the case that flow is considered to be safe it is sent back enclosed in $RC\,Reply$ flow otherwise it is dropped. Network devices don't implement cache to store flow content while waiting for $RC\,Reply$ and thus the whole flow needs to be forwarded although it means increase in network traffic. Forwarding all the traffic would cause excessive increase in network traffic and therefore each filter-for contains port number of traffic it relates to and forwarding is performed only for traffic containing this port number.

5 Experimental Results

As a testbed for our experiments we used an agent-based network simulation called Anet (Agent network). Several tests have been performed to evaluate the above-presented strategies and to identify their limitations. Scenario used for our experiments reported below contains 170 hosts of two different types. This means that each worm can infect approximately half of the population and the other half rises alarm upon receiving vulnerable flow. The propagation strategy that the worms are using to spread through the network is essential for our method - we have implemented a common strategy (similar to the one employed by Code Red II or Blaster worms) where worm is choosing IP address pseudo-randomly. There is 50% chance that the IP address will be from the same C-subnet as it's source address, and a 50% chance it will be from the same B-subnet. This strategy actually favors propagation on our simulated network, and is therefore a worst-case scenario – many worms scan local subnet with higher priority and than the whole IPv4 space.

In experimental scenario, we launch a sequence of worms into the system. Each worm starts spreading as soon as the last one has finished the spread, plus the time to terminate the reaction of the protective system. We observe the percentage of host that gets infected by worm as well as the percentage of malicious worm flows that are deleted by our filters before reaching their destination. These two parameters are related: more flows we filter, less hosts shall get infected, as we can clearly see in

Table 2. Percentage of infected hosts/filtered flows in experimental runs on identical network. Differences are due to the scanning strategy influence.

Experiment	**First worm**		**Second worm**	
Experiment	**% Filtered Flows**	**% Infected Hosts**	**% Filtered Flows**	**% Infected Hosts**
1	86%	09%	19%	93%
2	93%	07%	00%	98%
3	50%	22%	00%	00%
4	65%	51%	33%	54%
5	88%	01%	44%	86%
6	93%	01%	00%	85%
7	90%	03%	93%	28%
8	46%	39%	11%	68%
9	60%	07%	00%	51%
10	90%	13%	03%	30%
Avg	**76%**	**15%**	**23%**	**59%**

the results. On the other hand, the inherent randomness of the worm spread causes important variations in results presented in Table 2.

In the results, we can see that the system in its current stage of development is able to handle a single virus outbreak, and to spread its spread rather efficiently. The results clearly show that the system reacts only once the attack is reported by non-vulnerable hosts – it is not designed to prevent the outbreak, but to contain it and to limit its impact.

The results for second and all subsequent worms are rather similar – while the system is able to reduce the impact of the attack, its influence is limited and about 60% of vulnerable hosts are infected on average (compared to 15%) in case of the first worm. We attribute this degradation of performance to two factors - the detection module data is influenced by the activity of the first worm, and the capacity of network devices for filter deployment is already partially used.

6 Related Work

There are two fundamental approaches to network intrusion detection: *anomaly detection* and *signature detection*. Most commercial and existing open-source products fall into the *signature-detection* category: these systems match the packets in the network with the predefined set of rules [13]. While this system can be valuable in detecting known attack patterns, it provides no protection against novel threats that are not represented in its knowledge base. Furthermore, as the rules are typically sparse to achieve efficiency, the signature-based IDS have a non-negligible false positives ratio. Another major disadvantage of the existing systems is a high computational cost of reasonably complete set of rules – currently, complete SNORT database is not typically entirely deployed due to the computational power limitations.

The *anomaly detection* approaches in NIDS are typically based on classification/ aggregation of sessions or flows (unidirectional components of sessions) into classes and deciding whether a particular class contains malicious or legitimate flows. While

these approaches are able to detect novel attacks, they suffer from comparably high false-positive (or false negative depending on the model tuning) rate. Most research in the NIDS area currently delivers a combination of signature and anomaly detection and aims to improve the effectiveness and efficiency of intrusion detection. In the following paragraphs we are going to present a brief overview of selected (N)IDS systems [3]. Two of the recent representative systems are MINDS and SABER.

MINDS system [7] is an IDS system incorporating signature and anomaly detection suitable for host and network protection. As an input for traffic analysis it uses flow information which is aggregated in 10 minutes intervals. It extracts time-window and connection-window based features, from the traffic, to detect both fast and stealth attacks. The signature detection is used for protection against known threats. The anomaly detection the the principle of local outlier factor by which an unusual traffic is discovered. The human operator then decides which anomalous traffic is actually malicious and can use the captured information do define the signatures for subsequent traffic filtering.

SABER [14] combines both detection (anomaly and signature) and protection mechanisms. It incorporates a DoS-resistant network topology that shields valuable resources from the malicious traffic, and supports a business process migration to the safe location upon the attack. Real-time host-based IDS monitors program's use of Windows Registry and detects the actions of malicious software. The analogous approach has been used for file-based anomaly detection in Unix environment, when the IDS monitors a set of information about each file access. SABER is completed by autonomic software patching mechanism [15] which tries to automatically patch vulnerable software by identifying and correcting buffer overflows in the code, using the information from specialized honeypots operated alongside production machines.

7 Conclusions and Future Work

In our work, we present a multi-agent approach to intrusion detection and response, which relies on a combination of established IDS techniques (traffic observation, preprocessing and filtering) with efficient multi-agent approaches – trust modeling and distributed task allocation.

Our approach was evaluated in a high-level simulation of a campus-like network, featuring approx. 180 hosts and 30 network devices. Hosts were split in two groups, roughly of the same size, each with a different set of vulnerabilities. We have assumed that the network is protected by Firewall/NAT from direct attacks, and that the infections propagate from a single host (infected host connected by VPN, connected infected laptop, etc.). In this setting, the system is able to significantly reduce the spread of simulated worm, reducing the infection rate to approx. 20% of hosts infected before the attacks are reported, detected and the filter is crated and deployed on network devices.

In our current work, we concentrate on handling of multiple intrusions, prioritization of filtering and other issues where a rigorous application of multi-agent principles can tackle the real network-security problems.

References

1. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Internet quarantine: Requirements for containing self-propagating code. In: INFOCOM (2003)
2. Stallings, W.: Data and computer communications, 5th edn. Prentice-Hall, Inc., Englewood Cliffs (1997)
3. Axelsson, S.: Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ. (2000)
4. Rehak, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Klusch, M., Hindriks, K., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS(LNAI), vol. 4676, pp. 298-312. Springer, Heidelberg (2007)
5. Fischer, K., Muller, J.P., Pischel, M., Schier, D.: A model for cooperative transportation scheduling. In: Proceedings of the First International Conference on Multiagent Systems, pp. 109–116. AAAI Press / MIT Press, Menlo park, California (1995)
6. Maes, P.: Computational reflection. Technical report 87-2, Free University of Brussels, AI Lab (1987)
7. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J., Dokas, P.: Minds - minnesota intrusion detection system. In: Next Generation Data Mining, MIT Press, Cambridge (2004)
8. Rehak, M., Gregor, M., Pechoucek, M., Bradshaw, J.M.: Representing context for multiagent trust modeling. In: IAT'06. IEEE/WIC/ACM Intl. Conf. on Intelligent Agent Technology, USA, pp. 737–746. IEEE Computer Society, Los Alamitos (2006)
9. Haffner, P., Sen, S., Spatscheck, O., Wang, D.: Acas: automated construction of application signatures. In: MineNet '05. Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data, pp. 197–202. ACM Press, New York (2005)
10. Rehák, M., Foltýn, L., Pěchouček, M., Benda, P.: Trust model for open ubiquitous agent systems. In: Intelligent Agent Technology, 2005. IEEE/WIC/ACM International Conference, vol. PR2416, IEEE, Los Alamitos (2005)
11. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons, New York (2001)
12. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29, 1104–1113 (1980)
13. SNORT intrusion prevention system (2007) (accessed, January 2007), `http://www.snort.org/`
14. Keromytis, A.D., Parekh, J., Gross, P.N., Kaiser, G., Misra, V., Nieh, J., Rubenstein, D., Stolfo, S.: A holistic approach to service survivability. In: Proceedings of the 2003 ACM Workshop on Survivable and Self-Regenerative Systems (SSRS), pp. 11–22. ACM Press, New York (2003)
15. Sidiroglou, S., Keromytis, A.D.: Countering network worms through automatic patch generation. IEEE Security & Privacy 3, 41–49 (2005)

Agents Deliberating over Action Proposals Using the *ProCLAIM* Model

Pancho Tolchinsky[1], Katie Atkinson[2], Peter McBurney[2],
Sanjay Modgil[3], and Ulises Cortés[1]

[1]Knowledge Engineering & Machine Learning Group, Technical University of Catalonia, Spain
[2]Department of Computer Science, University of Liverpool, Liverpool, UK
[3]Advanced Computation Lab, Cancer Research UK

Abstract. In this paper we propose a dialogue game for agents to deliberate over a proposed action. The agents' dialogue moves are defined by a structured set of argument schemes and critical questions (CQs). Thus, a dialogue move is an instantiated scheme (*i.e.* an argument) or a CQ (*i.e.* a challenge on the argument instantiated in the scheme). The proposed dialogue game formalises the protocol based exchange of arguments defined in the *ProCLAIM* model. This model provides a setting for agents to deliberate over whether, given the arguments for and against, a proposed action is justified or not.

1 Introduction

Many domains illustrate requirements for multi-agent deliberation over a course of action (*e.g.*, the medical domain [12]). These requirements include the distributed nature of the data relevant to the decision making, the specialist nature of agents involved in the deliberative process, their geographical distribution and the need to maintain independence between these entities. However, the inevitable presence of uncertainty and disagreement amongst specialist agents suggests that any model of distributed reasoning should account for conflicts that arise during deliberation. Logical frameworks for argumentation based dialogue have in recent years emerged as one of the most promising paradigms for formalising reasoning of the above kind (*e.g.* [10,2]). Such frameworks provide a principled way in which to structure the exchange of, reasoning about, and evaluation of the arguments for proposals, between human and or automated agents. Exchanged arguments claiming beliefs and or proposals for action are evaluated on the basis of their conflict based interactions, and their relative strengths, in order to determine a preferred course of action. One of the most promising approaches is based on an argument schemes and critical questions approach (S & CQ) to argumentation over action [3]. Argument schemes, as described in the informal logic literature (e.g. [16]), are used to classify different types of argument that embody stereotypical patterns of reasoning. Instantiations of argument schemes can be seen as providing a justification in favour of the conclusion of the argument. The instantiated scheme (what we term an 'argument') can be questioned (attacked) through posing critical questions associated with the scheme. Each critical question can itself be posed as an attacking argument instantiating a particular scheme. This scheme is then itself subject to critical questioning. Computational accounts of the S & CQ approach to argumentation over action [4]

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 32–41, 2007.

have a number of advantages. The S & CQ effectively map out the *relevant* space of argumentation, in the sense that for any argument they identify the valid attacking arguments from amongst those that are logically possible. They also provide a natural basis for structuring argumentation based dialogue protocols. Indeed, protocols based on the S & CQ approach to argumentation over action [3] have been defined [5]. The latter work provides an abstract framework that can be instantiated and specialised for use in real world domains.

The primary contribution of this paper is that it proposes such a specialisation. A dialogue game for agents to argue over the appropriateness of an action is proposed for use within the *ProCLAIM* model (although the game is not inherently restricted to such use). This model posits the use of other components and knowledge sources required for practical and useful realisation of such a specialisation. In the following section we introduce the *ProCLAIM* model. In §3 we briefly describe one of the model's application scenarios, *viz.* human organ transplantation. In §4 we define the dialogue game. In §5 we present our intended future work, and in §6 we give our conclusions.

2 The *ProCLAIM* Model

The *ProCLAIM* model's primary component is a mediator agent (MA). Three tasks are defined for the MA: 1) The MA guides *proponent agents* as to what are their legal dialectical moves at each stage in a dialogue; 2) the MA also decides whether submitted arguments are valid (in the sense that instantiations of schemes are relevant w.r.t. the domain of discourse); 3) the MA evaluates the submitted valid arguments in order to provide an assessment of the appropriateness of the proposed action. In order to undertake these tasks, MA makes use of four knowledge resources, as shown diagrammatically in fig. 1) and also described below:

Argument Scheme Repository (ASR): In order to direct the proponent agents in the submission and exchange of arguments, the MA references a repository of argument schemes and their associated CQs. The schemes and CQs are instantiated by agents to construct arguments, and effectively encode the full 'space of argumentation', *i.e.*, all possible lines of reasoning that should be pursued w.r.t a given issue.

Guideline Knowledge (GK): This component enables the MA to check whether the arguments comply with the established knowledge, by checking what the valid instantiations of the schemes in ASR are (the ASR can thus be regarded as an abstraction of the GK). This is of particular importance in safety critical domains where one is under extra obligation to ensure that spurious instantiations of argument schemes should not bear on the outcome of any deliberation.

Case-Based Reasoning Engine (CBRe): This component enables the MA to assign strengths to the submitted arguments on the basis of their associated evidence gathered from past deliberations, as well as provide additional arguments deemed relevant in previous similar situations (see [15]).

Argument Source Manager (ASM): Depending on the source from whom the arguments are submitted, the strengths of these arguments may be readjusted by the MA. Thus, this component manages the knowledge related to, for example, the agents' roles and/or reputations.

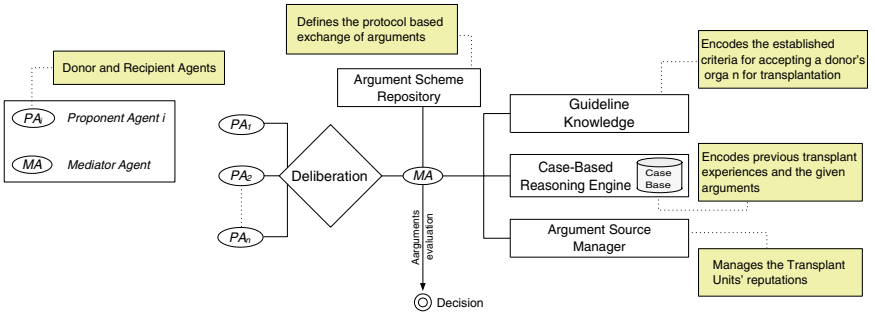


Fig. 1. *ProCLAIM*'s Architecture. Shaded boxes identify the model's constituent parts specialised for the transplant scenario introduced in §3.

The deliberation begins with one of the agents submitting an argument proposing an action. The MA will then guide the proponent agents in the submission of further arguments that will attack or defend the justification given for the proposed action. Each submitted argument instantiates a scheme of the ASR. The MA references the ASR in order to direct the proponent agents as to what their legal argument moves are at each stage of the deliberation. The strength[1] of each submitted argument is determined by the MA in referencing the other three knowledge resources, viz. GK, CBRe and ASM. Given all the accepted arguments, their strength and their interactions (based on the attack relation, see fig. 4b) the MA then applies Dung's seminal *calculus of opposition* [8] to determine the justified or *winning* arguments. Hence, if the initial argument is a winning argument, the proposed action is deemed appropriate, otherwise, it is rejected.

We now introduce one of our working scenarios, organ transplantation, in order to illustrate the use of *ProCLAIM*, and in particular the dialogue game.

3 The Transplant Scenario

The shortage of human organs for transplantation is a serious problem, and is exacerbated by the fact that current organ selection processes discard a significant number of organs deemed non-viable (not suitable) for transplantation. The organ viability assessment illustrates the ubiquity of disagreement and conflict of opinion in the medical domain. What may be a sufficient reason for discarding an organ for some qualified professionals may not be for others. Different policies in different hospitals and regions exist, and a consensus among medical professionals is not always feasible. Hence, contradictory conclusions may be derived from the same set of facts. For example, consider a donor with a smoking history but no *chronic obstructive pulmonary disease* (COPD). The medical guidelines indicate that a donor's smoking history is a sufficient reason for deeming a donor's lung as non-viable. However, there are qualified physicians that reason that the donor's lung is viable given that there is no history of COPD [9]. Similarly, the guidelines suggest discarding the kidney of a donor whose cause of death was

[1] Arguments deemed weaker than a given threshold are not accepted.

streptococcus viridans endocarditis (*sve*). However, by administrating *penicillin* to the recipient this means that the kidney can safely be transplanted.

Currently, the decision to offer for transplantation or discard an organ available for transplantation, is based solely on the assessment of doctors at the donor site (Donor Agent, DA). This organ selection process does not account for the fact that: *1)* medical doctors may disagree as to whether an organ is viable or non-viable; *2)* different policies in different hospitals and regions exist, and; *3)* viability is not an intrinsic property of the donor's organ, but rather, an integral concept that involves the donor and recipient characteristics as well as the courses of action to be undertaken in the transplantation process [9]. In particular, current organ selection process allow for a DA to discard an organ that medical doctors at the recipient site (Recipient Agents, RA) may claim to be viable and, given the chance, could provide strong arguments to support this claim.

In [13] a novel organ selection process is proposed in which *ProCLAIM* is used to coordinate joint deliberation between donor and recipient agents in order to prevent the discard of organs due to the application of inappropriate organ acceptability criteria. This helps reduce the disparity between the demand for and supply of organs.

ProCLAIM is thus instantiated with the proponent agents being the DA and RA, the Guideline Knowledge encodes the donor and organ acceptability criteria consented by the transplant organizations, i.e. the criteria the medical doctors should refer to when deciding the organs' viability.[2] The Argument Source Manager relates to the agents' reputation. Namely, the MA may deem as stronger the arguments submitted by agents with good reputation (*e.g.* a RA that has in the past successfully transplanted those organs which he claimed to be viable). Finally, the CBRe allows the MA to evaluate the submitted arguments on the basis of past recorded transplantation cases (see [15]).

In [14] a first attempt to formalise the ASR was presented, through a protocol-based exchange of arguments. However, argument schemes in that formalisation have to be constructed in a somewhat *ad-hoc* fashion, hindering the application of *ProCLAIM* in new scenarios (*e.g.* [6]). In the following section we introduce a simple dialogue game with six moves, corresponding to six argument schemes respectively. We intend to use these abstract scheme as a basis for the construction of the ASR (see future work).

4 Arguing over Action Proposals

In this section we introduce six abstract schemes that additionally represent six basic dialogue moves (later, in §5, we discuss how they can be extended). Our starting point is Atkinson's *et. al.* argument scheme over action [4]:

> In the current circumstances **R** we should perform action **A** to achieve new circumstances **S** which will realise some goal **G** which will promote some value **V**[3]

[2] Transplant organizations periodically publish the consented organ acceptability criteria. However, these criteria rapidly evolve because of the researchers' effort in extending them to reduce organ discards. Hence, the more advanced transplant units deviate from consented criteria.

[3] In this sense values represent the social interests promoted through achieving the goal. Thus they are qualitative, as opposed to quantitative, measures of the desirability of a goal.

Currently in *ProCLAIM* the appropriateness of an action is only evaluated with respect to one value as the agents are all reasoning so as to promote one value shared by all (*i.e.* the recipients' quality of life). Thus, the model does not take into account other possible values, such as the financial impact of the proposed action. Therefore, the value element of this argument scheme can be ignored for the purposes of this paper.

Another assumption of the model is that proposed actions are presumed to be appropriate, given some minimum context. Hence, to deem an action inappropriate agents must successfully argue that the proposed action has some undesirable side effect. That is, that the action realises some undesirable goal (in the transplant scenario, for example, these account for: `severe_infection`, `graft_failure`, `cancer`, *etc*).

Hence, agents *in favor* of the proposed action argue that NO undesirable goal will be realised. While, agents *against* the proposed action argue that an undesirable goal will be realised.[4] We formalise these proposals as follows:

Definition 1. An argument is represented in this dialogue game as a tuple:

$$< Context, Fact, Prop\_Action, Effect, Neg\_Goal >$$

Where **Context** is a set of facts that are not under dispute, that is, assumed to be true. **Fact** is a set of facts such that given the context $Context$, then the proposed action (or set of actions) **Prop_Action** result in a set of states **Effect** that realises some undesirable goal **Neg_Goal**. $Fact$ and $Effect$ may be empty sets and Neg_Goal may be equal to `nil`, representing that no undesirable goal is realised. So, arguments in favor of a proposed action are of the form: $< Context, Fact, Prop\_Action, Effect,$ `nil` $>$ whereas arguments against a proposed action, for instance against a transplantation of an organ, highlight some negative goal that will be realised: *e.g.*$< Context, Fact, Prop\_Action, Effect,$`severe_infection` $>$

Hence, the arguments used in the dialogue take into account: *1)* the current state of affairs referenced by the facts deemed relevant by the proponent agents; *2)* the proposed actions; *3)* the new state achieved if a proposed action is undertaken, and; *3)* the undesirable goals which the new state realises.

In *ProCLAIM*, a proposed action (*e.g.* transplant an organ) is deemed appropriate if there are no expected undesirable effects. Thus, a proposed action is by default assumed appropriate. Nonetheless, there must be some minimum set of conditions for proposing such an action (*e.g.* an available organ and a potential recipient for that organ). Thus, the dialogue starts by submitting an argument that claims the appropriateness of an action and the subsequent dialogue moves will attack or defend the presumptions present in that argument by claiming there is (resp. there is not) an undesirable side effect.

The six schemes we now introduce are partial instantiation of the more general scheme introduced in definition 1. These more specific schemes are intended to help identify the legal instantiation of the more general scheme at each stage of the dialogue.

[4] Other accounts making use of undesirable outcomes (goals to be avoided) exist in the literature, e.g. Bench-Capon and Prakken's account [7], which is focused on the concept of accrual, and Amgoud and Kaci's account [1], which is defined for use in negotiation dialogues. The aims of these two accounts differ from ours, thus we will make no further use of them in this paper.

Let us consider R and S to be finite sets of facts in the current state and the resultant state respectively. Let A be a finite set of actions and G^- a finite set of undesirable, or negative, goals ($\texttt{nil} \in G^-$). Let the elements of R, S and A be represented as predicates with grounded variables (*e.g.* $\texttt{donor(john,lung)} \in R$). Let, also, elements in G^- be represented as propositions (*e.g.* $\texttt{severe\_infection}$).

Thus, a dialogue starts with the submission of the argument:

AS1: $< m\_c, \{\}, p\_a, \{\}, \texttt{nil} >$

Where $m\_c \subseteq R$ is a minimum set of facts that an agent requires for proposing a non-empty set of actions $p\_a \subseteq A$. For example, if a lung of a donor d ($\texttt{donor(d,lung)}$) is available for a potential recipient r ($\texttt{pot\_recip(r,lung)}$), the argument for transplantation ($\texttt{transp(r,lung)}$) instantiates AS1 as follows:

$A = < \{\texttt{donor(d,lung)}, \texttt{pot\_recip(r,lung)}\}, \{\}, \{\texttt{transp(r,lung)}\}, \{\}, \texttt{nil} >$

Associated with the argument scheme are CQs that question the justification presented in an instantiated argument. In particular, to $AS1$ is associated the CQs[5]:

AS1_CQ1: Is there a contraindication for undertaking the proposed action?

The CQs identify the following dialogue moves by instantiating argument schemes that correspond to the posing of the CQs. Thus, associated with AS1_CQ1 is the scheme:

AS2: $< m\_c, r1, p\_a, s1, g^- >$, where $r1 \subset R$, $s1 \subset S$ and $g1^- \in G^-/\{\texttt{nil}\}$ ($r1$ and $s1$ nonempty). That is, **AS2** identifies contraindications $r1$ for undertaking p_a.

For relevance and consistency we require that the facts in $r1$ are non redundant nor in conflict with those in m_c. That is, that $r1$ is *consistently relevant* w.r.t to m_c.

Definition 2. Let B be a nonempty set, then we say that B is *consistently relevant* w.r.t the set A, iff: given a background theory Γ with a consequence relation $\vdash$ then:

1) $\forall b \in B, \quad A \nvdash_\Gamma b$ and *2)* $A \cup B \nvdash_\Gamma \bot$

In the simplest case ($\Gamma = \emptyset$), this amounts to $b \notin A$ and $\neg b \notin A, \forall b \in B$.

Let us suppose that the donor of the offered lung has *smoking history* ($\texttt{d\_p(d,s\_h)}$). Let us suppose as well that the DA believes s_h is a contraindication because it may lead to a graft failure. The agent can submit an argument $B1$ attacking A by instantiating AS2 as follows (see fig. 3, Example 1): r1 = $\{\texttt{d\_p(d,s\_h)}\}$; s1 = $\{\texttt{reject(r,lung)}\}$; $g1^- = \{\texttt{graft\_failure}\}$.

Associated with AS2 are three CQs that in turn identify three argument schemes:

AS2_CQ1: Are the current circumstances such that the stated effect will be achieved?

[5] The CQs associated with the scheme enable agents to attack the validity of the various elements of the scheme and the connections between them. Also, there may be alternative possible actions and side effects of the proposed action. Only a subset of the CQs for arguing over action proposals are relevant in this particular application. The full list can be found in [4].

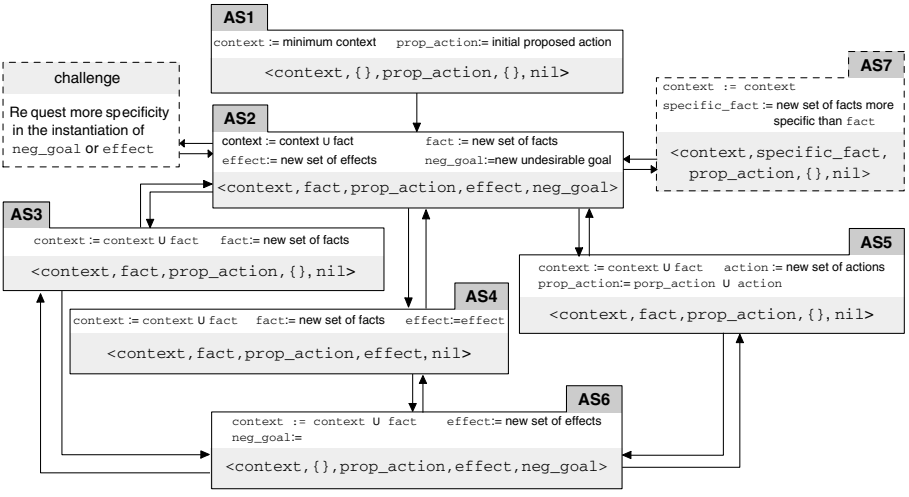


Fig. 2. Schematic representation of the dialogue game

AS3: $< m\_c \bigcup r1, r2, p\_a, \{\}, \texttt{nil} >$, where $r2 \subset R$ is *consistently relevant* w.r.t. the updated context $m\_c \bigcup r1$.

AS2_CQ2: Are the current circumstances such that the achieved effect will realise the stated goal?

AS4: $< m\_c \bigcup r1, r2, p\_a, s1, \texttt{nil} >$, where $r2 \subset R$ is *consistently relevant* w.r.t. the updated context $m\_c \bigcup r1$.

AS2_CQ3: Is there a course of action that prevents the achievement of the stated effect?

AS5: $< m\_c \bigcup r1, \{\}, p\_a \bigcup a, s1, \texttt{nil} >$, where $a \subset A$.

Supposing the RA believes smoking history is not a contraindication because the donor did not have chronic obstructive pulmonary disease (*codp*), by instantiating scheme AS3 with $r2 = \{\neg \texttt{d\_p(d, copd)}\}$ RA can submit argument $C1$ (see fig. 3) that attacks $B1$ (see fig. 4b). Figure 2 depicts a schematic representation of the dialogue game.

Arguments instantiating schemes **AS3**, **AS4** or **AS5**, are of the form $< context, fact, prop\_act, effect, \texttt{nil} >$, with $fact$ and $effect$ possibly empty. These arguments introduce either a new fact or a new "prophylactic" action that may in turn warrant, respectively cause, some undesirable secondary effect. Hence, associated with these schemes is the CQ:

AS3,4,5_CQ1: Will the introduced factor cause some undesirable side effects?

AS6: $< context \bigcup fact, \{\}, prop\_action, s2, g2^{-} >$, with $s2 \subseteq S$, nonempty, and $\exists \alpha \in s2$ such that $\alpha \notin s1$ (see fig. 3 Example 2, for an illustration).

An argument instantiating **AS6** can in turn be attacked by arguments instantiating **AS3**, **AS4** or **AS5**. Also, an argument instantiating **AS3**, **AS4** or **AS5**, alter either the

	ID	Type	Argument
Exampl 1	A	AS1	<{donor(d,lung), pot_recip(r,lung)}, {}, {}, transp(r,lung), {}, nil >
	B1	AS2	<{donor(d,lung), pot_recip(r,lung)}, {d_p(d,s_h)}, prop_act, {reject(r,lung)}, graft_failure>
	C1	AS3	<{donor(d,lung), pot_recip(r,lung)} U {d_p(d,s_h)}, {d_p(d,no_copd)}, prop_act, {}, nil>
Exampl 2	B2	AS2	<{donor(d,lung), pot_recip(r,lung)}, {d_p(d,hiv)}, prop_act, {recip_p(r,hiv)}, sever_infect>
	C2	AS4	<context U {d_p(d,hiv)}, {p_r_p(r,hcv)}, prop_act, {rec_p(r,hiv)}, nil >
	D2	AS6	<{donor(d,lung), pot_recip(r,lung)} U {p_r_p(r,hiv)}},{}, prop_act, {recip_p(r,super_infect)}, sever_infect>
Exampl 3	B3	AS2	<{donor(d,lung), pot_recip(r,lung)}, {d_p(d,sve)}, prop_act, {recip_p(r,svi)}, sever_infect>
	C3	AS5	<{donor(d,lung), pot_recip(r,lung)} U {d_p(d,ve)}, {}, prop_act U {treat(r,penicillin)}, {}, nil >
	D3	AS2	<{donor(d,lung), pot_recip(r,lung)}, {p_r_p(r,pen_allergy)}, prop_act, {recip_p(r,anaphylaxis)}, sever_infect>

Fig. 3. Three example dialogues resulting from the submission of an argument A proposing transplantion of a lung (`transp(r,lung)`) to a potential recipient r (`pot_recip(r,lung)`). Figure 4a. depicts the argument interaction. **Example 1**: Argument $B1$ attacks A by identifying smoking history, a property of the donor (`d_p(d,s_h)`), as a contraindication, causing an eventual rejection in the recipient(`reject(r,lung)`) that realises the negative goal of graft failure. However, argument $C1$ attacks $B1$ claiming that this consequence will not be achieved because the donor does not have a chronic obstructive pulmonary disease ($\neg$`d_p(d,copd)`). **Example 2**: Argument $B2$ identifies HIV as a contraindication (`d_p(d,hiv)`) that causes the recipient to also develop HIV, a severe infection. $C2$ claims that the potential recipient already has HIV (`p_r_p(r,hiv)`) and so for that recipient, HIV will not be a severe infection. Argument $D2$, attacks $C2$ by stating that if the donor and potential recipient have HIV, a consequence of the transplant is a superinfection. Finally, in **example 3**, argument $B3$ identifies the contraindication *streptococcus viridans endocarditis* in the donor (`d_p(d,sve)`) which will result in a *streptococcus viridans infection* in the recipient (`r_p(r,svi)`). Argument $C3$ states that this infection can be prevented if the recipient is treated with penicillin (`treat(r,penicillin)`). However, argument $D3$ attacks the appropriateness of the action by stating that the potential recipient is allergic to this antibiotic (`p_r_p(r,pen_allergy)`), thus causing anaphylaxis in the recipient (`r_p(r,anaphylaxis)`).

context or *proposed action* set. Hence, as depicted in figure 2 this argument can be attacked by an argument instantiating **AS2**.

A submitted argument in the dialogue attacks the argument it replies to. However, some arguments, in particular, those that claim the proposed action realises some undesirable goal (*i.e.* arguments instantiating **AS2** and **AS6**) will counter-attack an attack made on the argument (*i.e.*, the arguments symmetrically attack). Hence, the nature of the attack between arguments in the dialogue differs depending on the argument schemes they instantiate (see fig. 4a). Broadly speaking, the rationale as to whether argument A symmetrically or asymmetrically attacks B, is that in the latter case B is an argument in favor of a proposed action that makes an assumption that there is no contraindication. A contradicts this assumption. In the former symmetric case, there is a disagreement (it is a moot point) as to whether the newly introduced factor is or is not a contraindication for the proposed action (see [11] for more details on the motivation).

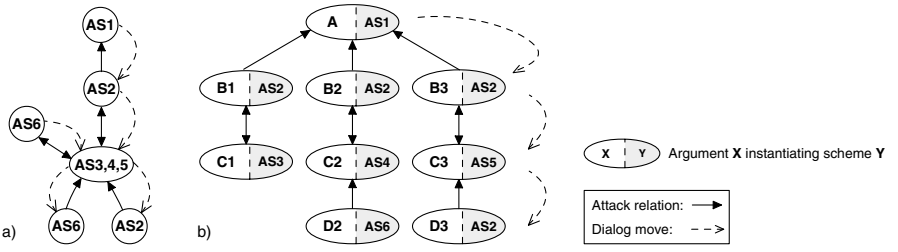


Fig. 4. a) Attack relation between the Argument Schemes. b) Dialogue graph illustrating the interaction between the arguments used in the three examples given in figure 3.

5 Future Work

In domains with uncertainty agents may not always be able to provide a precise account of the expected effects of an action and thus, the negative goal these may realise. Thus agents may require terms with different levels of specificity. However, if the specificity relation is introduced in R, S, A and G^-, new argument-based dialogue moves become relevant. For example, in figure 2 two additional dialogue moves were proposed: *1)* A **challenge** move that requires agents to be more specific on the terms used. *2)* If a fact (*e.g. cancer* on the donor) is identified as causing an undesirable effect (*cancer* on the recipient) via scheme AS2, in order to attack this argument the scheme **AS7** can be instantiated with a more specific account of the introduced fact (*e.g. non systemic cancer*) which does not cause the stated undesirable effect (although cancer is in general a contraindication, the donor has a cancer which is not systemic, where the latter, more specific, is not a contraindication).

We also intend to extend the dialogue game so as to allow for arguing over beliefs. In particular, to question whether an introduced fact is indeed the case (*e.g.* does the donor actually have a smoking history?).

Currently we are working on a methodology for constructing the *ProCLAIM*'s Argument Scheme Repository. The idea is to further specialise the introduced argument schemes to construct a repository of domain dependent schemes.

6 Conclusions

In this paper we have proposed a dialogue game for deliberating over action proposals. This dialogue game is intended to (although not restricted to) 1) formalise the protocol-based exchange of arguments in *ProCLAIM* ; *2)* provide a basis for the construction of the Argument Scheme Repository. The dialogue can be regarded as a specialisation of that proposed by Atkinson *et al*. in [4]. The specialisation is required in order provide an implementation that ensures that all the relevant factors can be elicited from the proponent agents [14] in order to decide whether it is safe to undertake the proposed action, taking into account that the dialogue participants may disagree.

The *ProCLAIM* model provides for practical realisation of the dialogue game by referencing knowledge sources that describe the schemes and critical questions, ensure that instantiations of the schemes are valid, and bring to bear the results of previous

dialogues (cases) and agent reputations in order that the relative strengths of arguments exchanged during the dialogue can be evaluated.

Acknowledgments. This research was supported in part by EC Project ASPIC (FP6-IST-002307).

References

1. Amgoud, L., Kaci, S.: On the generation of bipolar goals in argumentation-based negotiation. In: Rahwan, I., Moraïtis, P., Reed, C. (eds.) ArgMAS 2004. LNCS (LNAI), vol. 3366, pp. 192–207. Springer, Heidelberg (2005)
2. ASPIC: Deliverable d2.1: Theoretical frameworks for argumentation (June 2004), `http://www.argumentation.org/Public_Deliverables.htm`
3. Atkinson, K.: What Should We Do?:Computational Representation of Persuasive Argument in Practical Reasoning. PhD thesis, Department of Computer Science, University of Liverpool, UK (2005)
4. Atkinson, K., Bench-Capon, T., McBurney, P.: Computational representation of practical argument. Synthese 152(2), 157–206
5. Atkinson, K., Bench-Capon, T., McBurney, P.: A dialogue game protocol for multi-agent argument over proposals for action. Autonomous Agents and Multi-Agent Systems 11, 153–171 (2005)
6. Aulinas, M., Tolchinsky, P., Turon, C., Poch, M., Cortés, U.: Is my spill environmentally safe? towards an integrated management of wastewater in a river basin using agents that can argue. In: WATERMATEX (May 2007)
7. Bench-Capon, T., Prakken, H.: Choosing what to do by accruing arguments. In: Conference on Computational Models of Natural Argument. Frontiers in Artificial Intelligence and Applications, vol. 144, pp. 247–258. IOS Press, Amsterdam (2006)
8. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77, 321–357 (1995)
9. López-Navidad, A., Caballero, F.: Extended criteria for organ acceptance: Strategies for achieving organ safety and for increasing organ pool. Clin Transplant, Blackwell Munksgaard 17, 308–324 (2003)
10. McBurney, P., Parsons, S.: Dialogue game protocols. In: Communication in Multiagent Systems, pp. 269–283 (2003)
11. Modgil, S., Tolchinsky, P., Cortés, U.: Towards formalising agent argumentation over the viability of human organs for transplantation. In: Mexican International Conference on Artificial Intelligence 2005, pp. 928–938 (2005)
12. Nealon, J., Moreno, A.: The application of agent technology to healthcare. In: 1st International Joint Conference on Autonomous Agents and Multiagent Systems (2002)
13. Tolchinsky, P., Cortés, U., Modgil, S., Caballero, F., López-Navidad, A.: Increasing human-organ transplant availability: Argumentation-based agent deliberation. IEEE Intelligent Systems 21(6), 30–37 (2006)
14. Tolchinsky, P., Modgil, S., Cortés, U.: Argument schemes and critical questions for heterogeneous agents to argue over the viability of a human organ. In: AAAI 2006 Spring Symposium Series; Argumentation for Consumers of Healthcare, pp. 105–111. AAAI Press, Stanford, California, USA (2006)
15. Tolchinsky, P., Modgil, S., Cortés, U., Sànchez-Marrè, M.: CBR and Argument Schemes for Collaborative Decision Making. In: Conference on Computational Models of Natural Argument, vol. 144, pp. 247–258. IOS Press, Amsterdam (2006)
16. Walton, D.N.: Argument Schemes for Presumptive Reasoning. Lawrence Erlbaum Associates, Mahwah, NJ, USA (1996)

An Attacker Model for Normative Multi-agent Systems

Guido Boella[1] and Leendert van der Torre[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino, Italy
guido@di.unito.it
[2] Computer Science and Communication (CSC), University of Luxembourg
leendert@vandertorre.com

Abstract. In this paper we introduce a formal attacker model for normative multi-agent systems. In this context, an attacker is an agent trying to profit from norm violation, for example because the violation is not detected, it is not being sanctioned, or the sanction is less than the profit of violation. To deliberate about norm violations, an attacker has a self model and a model of the normative multi-agent system, which in our case have the same structure. Moreover, we assume that an attacker violates a norm only when it profits from it, and the attacker therefore plays a violation game with the system. On a variety of examples, we show also how our model of violation games based also on agent abilities or power extends our earlier model based on motivations only.

1 Introduction

To study the security of a normative multi-agent system [3], we have to build a model of an attacker or adversary. Though explicit attacker models are well known and studied in cryptography and computer security, it seems that the concept has not been developed thus far for normative multi-agent systems.

In this context, an attacker is an agent trying to profit from norm violation, for example because the violation is not detected, it is not being sanctioned, or the sanction is less than the profit of violation [1]. Attacker models extend our earlier violation games studying fraud and deception [2], because these games consider only the agents' decision variables directly controlled by the agents, without modelling the preconditions of the decisions or their consequences.

In this paper we show that if we consider also the agent's ability or power to change the state of the world via conditional effects of decisions, we can model new interesting violation games. In particular, the attacker can exploit indirect conflicts between effects of decision variables, and incompatible decisions.

The layout of this paper is as follows. We first introduce our earlier model by describing the range of violation games that can be played in it, together with various violation games involving an attacker that are not covered by the previous model. Then we introduce our new formal attacker model, and finally we illustrate the model by formalizing some of the examples.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 42–51, 2007.

2 Violation Games of Attackers of Normative Systems

Normally an agent fulfills its obligations, because otherwise its behavior counts as a violation that is sanctioned, and the agent dislikes sanctions [1]. Moreover, it may dislike violations regardless of sanctions, and it may act according to the norm regardless whether its behavior counts as a violation. In this section we discuss four main exceptions to this normal behavior, the former two taken from [2] are related to motivations only, the latter two are dealing also with abilities or powers of agents.

First, an agent may violate an obligation when the violation is preferred to the sanction, like people who like to speed and do not care about the speeding tickets. In such cases the system may increase its sanctions. An instance occurs when the agent is sanctioned and does not care about additional sanctions. In this case the additional sanction may not be high enough, but often the first sanction was too high. An example is the famous Beccaria's argument against death penalty: it makes sanctions associated to other crimes irrelevant.

Moreover, the agent may have conflicting desires or obligations which it considers more important. First, obligations may conflict with each other. E.g., if the normative system for web access is incoherent it may forbid and oblige at the same time to access a robot.txt file. Second, obligations may conflict with the agent's private preferences, since they are adopted as goals by agents who respect them. The agent can desire exactly the opposite it is obliged to, e.g., in a file sharing system a record company prefers not to share its music files. In the case of conflict with goals, the matter becomes more complex: besides deciding whether the obligation is preferred to the goal, the agent must consider whether it should change its mind. Different types of agents can have different attitudes towards goal reconsideration. Third, if an agent does not fulfill an obligation, then there is a conflict between the sanction it is subject to and its desires and goals, since by definition the sanction must not be preferred. E.g., the sanction can be the inability to access a resource or the loss of a monetary deposit.

Second, the agent may think that its behavior will not be counted as a violation, or that it will not be sanctioned. This may be the case in the normal situation when the agents working for the normative system are lazy, because the sanction has a cost for the system, but more likely it is due to an action of the agent. This action may abort the goal of the normative system to count the agent's behavior as a violation or to sanction it, as in the case of bribing, or it may trigger a conflict for the normative system. An agent can use recursive modelling [6] to exploit desires and goals of the normative system thus modifying its motivations and inducing it not to sanction. In particular, the agent can work on the conditional desires and goals [5] of the system by triggering some of them which conflict with the goal of sanctioning the agent. We distinguish two situations. In the first case, the normative system has no advantage from the action executed by the agent. E.g., in some countries members of parliament cannot be sanctioned, so an agent can become an MP in order not to be punished. In the second case, the normative system gains an advantage from the action of the agent. A particular case of this behavior is bribing: some agents working for the

normative system have a disposition to be bribed, i.e., they have a goal towards not sanctioning the agent that can be triggered by a payment by part of the agent; if the payment to the system is smaller than the cost of disk space, then the agent profits by bribing agents working for the system.

Third, no motivation can lead the agent to fulfill an obligation if it cannot achieve it: e.g., in a file sharing system the agent may have already reached its quota of disk space, so it has no space left to put at disposal of the community it belongs to. Moreover, the difference with the examples in [2] is that with abilities there is not necessarily an explicit conflict in the obligations posed by the normative system: the agent may not have enough resources to fulfill all the obligations, or the actions for fulfilling the obligations have incompatible effects. Conflicts may arise indirectly also with actions the agent desires to perform, for example, it cannot use the disk space for installing software it needs.

Fourth, the normative system can be unable to count the behavior as a violation or sanction it. This behavior is caused by an action of the agent, either by blocking the sanction directly, or creating a conflict with other obligations. Concerning influencing the normative system, the explicit modelling of the system's abilities or power allows more ways for the agent to violate obligations while avoiding the sanction. The recognition of a violation and the sanction may require some applicability conditions to be achieved. Hence the agent can manipulate these conditions: it can make it impossible for the system to prosecute it or to perform the sanction. E.g., an access control system is not able anymore to block the agent's connections, since it has changed its IP address. Alternatively, the agent can make it more difficult, and hence more costly, for the system to execute the prosecution process and the sanction. E.g., some of the agent's actions can in fact 'trigger' some side effect of the action of sanctioning. The agent can use proxy servers to connect to some access control system, so that it is more difficult to block the agent's connections. This additional cost may make it not worthwhile to enforce the respect of the obligation.

Moreover, another way of changing the system's behavior is to trigger some of its other goals, so that the system is led to plan a solution for achieving also those goals. Which goal is useful to trigger depends on which action the system is compelled to execute to achieve the new goal, if the system prefers to achieve that goal with respect to sanctioning. For example, the agent could launch a denial of service attack against the system. Since for the system it is more important to stop the attack than to apply the sanction and it cannot do both actions, the system drops its decision to sanction the agent. A very particular case of this situation is due to the fact that, in our model, sanctions are modelled as conditional goals which are triggered by a violation. So, paradoxically, the agent can trigger the system's action by violating some other obligation whose sanction makes the normative system want not to apply further sanctions. If this new sanction is less harsh then the first one and the system is compelled to punish only the second obligation first, e.g., the goal to apply the first sanction prevails to the goal of applying the second one, then the agent can even benefit from violating two obligations while being sanctioned for only one.

3 Attacker Model for Normative Multiagent Systems

To deliberate about norm violations, an attacker model consists of:

A self model of the attacker, distinguishing, for example, between norm internalizing agents, respectful agents and selfish agents. In this paper we use a Belief-Desire-Goal model to represent the attacker's mental attitudes.

A model of the normative multi-agent system, where for efficiency reasons we assume that the attacker describes the normative multi-agent system with the same structures used to describe its self-model. The beliefs, desires and goals of the system reflect these mental attitudes of agents working for it, such as legislators, judges, and policemen.

A model of interaction, where the agent plays a so-called violation game with the system, based on recursive modeling [6].

In the definition of multiagent system, not necessarily all variables are assigned to an agent, and we introduce effect rules. The description of the world contains – besides decision variables whose truth value is determined directly by an agent – also *parameters* whose truth value can only be determined indirectly. The distinction between decision variables and parameters is a fundamental principle in all decision theories or decision logics [4,7]. To distinguish the model developed in [2] from the one developed in this paper, we call the model in this paper our second normative multi-agent system.

Definition 1 (Agent description). *The tuple* $\langle A, X, D, G, AD, E, MD, \geq \rangle$ *is our second multiagent system* MAS_2*, where*

- *the agents* A*, variables* X*, desires* D *and goals* G *are four finite disjoint sets. We write* $M = D \cup G$ *for the motivations defined as the union of the desires and goals.*
- *an agent description* $AD : A \to 2^{X \cup D \cup G}$ *is a complete function that maps each agent to sets of decision variables, desires and goals. For each agent* $a \in A$*, we write* X_a *for* $X \cap AD(a)$*,* D_a *for* $D \cap AD(a)$*,* G_a *for* $G \cap AD(a)$*. We write parameters* $P = X \setminus \cup_{a \in A} X_a$*.*
- *the set of literals built from* X*, written as* $L(X)$*, is* $X \cup \{\neg x \mid x \in X\}$*, and the set of rules built from* X*, written as* $R(X) = 2^{L(X)} \times X$*, is the set of pairs of a set of literals built from* X *and a literal built from* X*, written as* $\{l_1, \ldots, l_n\} \to l$*. We also write* $l_1 \wedge \ldots \wedge l_n \to l$ *and when* $n = 0$ *we write* $\top \to l$*. Moreover, for* $x \in X$ *we write* $\sim x$ *for* $\neg x$ *and* $\sim(\neg x)$ *for* x*.*
- *the set of effects* $E \subseteq R(X)$ *is a set of rules built from* X*.*
- *the motivational description* $MD : M \to R(X)$ *is a complete function from the sets of desires and goals to the set of rules built from* X*. For a set of motivations* $S \subseteq M$*, we write* $MD(S) = \{MD(s) \mid s \in S\}$*.*
- *a priority relation* $\geq$*:* $A \to 2^M \times 2^M$ *is a function from agents to a transitive and reflexive relation on the powerset of the motivations containing at least the subset relation. We write* $\geq_a$ *for* $\geq (a)$*.*

We model the attacker and the normative multiagent system as two agents **a** and **n**, such that we can describe the interaction between the attacker and the system as a game between two agents. In the definition of normative multiagent system, we add the fact that violations can be mapped on parameters as well as decision variables of system **n**, which formalizes the property that in some circumstances system **n** may not be able to count behavior as violation. Moreover, we change the definition of goal distribution such that agents can also be responsible for parameters.

Definition 2 (Norm description). *Let $\langle A, X, D, G, AD, E, MD, \geq\rangle$ be our second multiagent system. The tuple $\langle A, X, D, G, AD, E, MD, \geq, \mathbf{n}, N, V, GD\rangle$ is our second normative multiagent system $NMAS_2$, where:*

- *the normative system $\mathbf{n} \in A$ is an agent.*
- *the norms $N = \{n_1, \ldots, n_m\}$ is a set disjoint from A, X, D, and G.*
- *the norm description $V : N \times A \rightarrow X_{\mathbf{n}} \cup P$ is a complete function from the norms to the decision variables of the normative agent together with the parameters: we write $V(n, a)$ for the decision variable which represents that there is a violation of norm n by agent $a \in A$.*
- *the goal distribution $GD : A \rightarrow 2^{G_{\mathbf{n}}}$ is a function from the agents to the powerset of the goals of the normative agent, where $GD(a) \subseteq G_{\mathbf{n}}$ represents the goals of agent **n** the agent a is responsible for, such that if we have $L \rightarrow l \in MD(GD(a))$, then $l \in L(X_{\mathbf{a}} \cup P)$.*

4 Obligations in the Normative Multi-agent System

We introduce a logic of rules *out* for the desires and goals of the agents, and a second logic of rules for the effect rules, called *outE*. Both take the transitive closure of a set of rules, called reusable input/output logic in [8], but the latter also includes the input in the output.

Definition 3 (Out). *Let $MAS_2 = \langle A, X, D, G, AD, E, MD, \geq\rangle$ be our second multiagent system. Moreover, let:*

- *$out(D, S)$ be the closure of $S \subseteq L(X)$ under the rules $D \subseteq R(X)$:*
  - *$out^0(D, S) = \emptyset$*
  - *$out^{i+1}(D, S) = out^i(D, S) \cup \{l \mid L \rightarrow l \in D, L \subseteq out^i(D, S) \cup S\}$ for $i \geq 0$*
  - *$out(D, S) = \cup_0^\infty out(D, S)$*

  Moreover, following notational conventions in input/output logics, we write $a \rightarrow x \in out(R)$ for $x \in out(R, \{a\})$.
- *$outE(E, S)$ be the closure of $S \subseteq L(X)$ under the effect rules E:*
  - *$outE^0(E, S) = S$*
  - *$outE^{i+1}(E, S) = outE^i(E, S) \cup \{l \mid L \rightarrow l \in E, L \subseteq outE^i(E, S)\}$ for $i \geq 0$*
  - *$outE(E, S) = \cup_0^\infty outE(E, S)$*

  Moreover, we write $a \rightarrow x \in outE(R)$ for $x \in outE(R, \{a\})$.

Example 1. Consider the multiagent system $\langle A, X, D, G, AD, E, MD, \geq \rangle$ where $A = \{\mathbf{a}, \mathbf{n}\}$, $X_{\mathbf{a}} = \{x\}$, $X_{\mathbf{n}} = \{y\}$. Agent $\mathbf{a}$ desires and wants unconditionally to decide to do x, but if agent $\mathbf{n}$ decides y, then it wants the opposite $\neg x$: $MD(D_{\mathbf{a}}) = MD(G_{\mathbf{a}}) = \{\top \rightarrow x, y \rightarrow \neg x\}$. The second rule is preferred over the first one, the ordering $\geq_{\mathbf{a}}$ is $\{\top \rightarrow x, y \rightarrow \neg x\} > \{y \rightarrow \neg x\} > \{\top \rightarrow x\} > \emptyset$. System $\mathbf{n}$ desires and wants to do y: $MD(D_{\mathbf{n}}) = MD(G_{\mathbf{n}}) = \{\top \rightarrow y\}$. Moreover, assume $P = \{p, q\}$ and $E = \{\top \rightarrow p, x \rightarrow q, y \rightarrow \neg q\}$. The output is given in Figure 1. The unconditional effect rule $\top \rightarrow p$ represents what is true in the initial state. $outE(E, \{x\}) = \{x, p, q\}$, while $outE(E, \{y\}) = \{x, p, \neg q\}$. The remainder of this figure is explained in the following section.

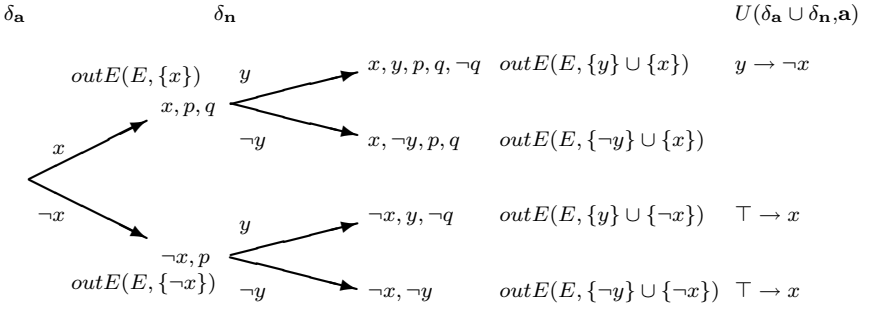


Fig. 1. Violation game tree

The obligations, count-as-violations as well as sanctions can be parameters, which can only be achieved indirectly by the agents' decisions. For count-as-violations and sanctions, we extend the definition of obligation with the additional clause that the normative agent has at least one way to apply the sanction. To formalize this clause, we already have to define what decisions are (which in [2] we could delay until the section on behavior).

Definition 4 (Decisions). *The set of decisions Δ is the set of subsets $\delta \subseteq L(X \setminus P)$ such that their closure under effect rules $outE(E, \delta)$ does not contain a variable and its negation. For an agent $a \in A$ and a decision $\delta \in \Delta$ we write δ_a for $\delta \cap L(X_a)$.*

Given decisions we define the ability of an agent to make true a propositional variable in a certain context by means of a decision:

Definition 5 (Ability). *Agent $a \in A$ is able to achieve $p \in L(X)$ in context $Y \subseteq L(X)$, written as $able(a, p, Y)$, if and only if there is a decision δ_a such that $p \in outE(E, Y \cup \delta_a)$.*

In the following section we show that for our recursive games the new clauses do not imply that the normative agent can always count behavior as a violation and sanction it. The reason is that some decisions of the normative multi-agent system can be blocked due to decisions of agent $\mathbf{a}$. E.g., if agent $\mathbf{a}$ sees to it that

a parameter p is true, then all decisions of system $\mathbf{n}$ are blocked that would see to $\neg p$, because the effects of decision must be consistent.

Definition 6 (Obligation). *Let*

$$NMAS_2 = \langle A, X, D, G, AD, E, MD, \geq, \mathbf{n}, N, V, GD \rangle$$

be our second normative multiagent system. In $NMAS_2$ agent $\mathbf{a} \in A$ is obliged to decide to do $x \in L(X_\mathbf{a} \cup P)$ with sanction $s \in L(X_\mathbf{n} \cup P)$ if $Y \subseteq L(X_\mathbf{a} \cup P)$, written as $NMAS_2 \models O_\mathbf{an}(x, s|Y)$, if and only if $\exists n \in N$ such that:

1. $Y \rightarrow x \in out(D_\mathbf{n}) \cap out(GD(\mathbf{a}))$*: if Y then system $\mathbf{n}$ desires and has as a goal that x, and this goal has been distributed to agent $\mathbf{a}$.*
2. $Y \cup \{\sim x\} \rightarrow V(n, \mathbf{a}) \in out(D_\mathbf{n}) \cap out(G_\mathbf{n})$*: if Y and $\sim x$ is done by agent $\mathbf{a}$, then system $\mathbf{n}$ has the goal and the desire $V(n, \mathbf{a})$: to recognize it as a violation done by agent $\mathbf{a}$.*
3. $\top \rightarrow \neg V(n, \mathbf{a}) \in out(D_\mathbf{n})$*: system $\mathbf{n}$ desires that there are no violations.*
4. $Y \cup \{V(n, \mathbf{a})\} \rightarrow s \in out(D_\mathbf{n}) \cap out(G_\mathbf{n})$*: if Y and system $\mathbf{n}$ decides $V(n, \mathbf{a})$ then system $\mathbf{n}$ desires and has as a goal that it sanctions agent $\mathbf{a}$.*
5. $Y \rightarrow \sim s \in out(D_\mathbf{n})$*: system $\mathbf{n}$ desires not to sanction, $\sim s$. This desire of the normative system expresses that it only sanctions in case of a violation.*
6. $Y \rightarrow \sim s \in out(D_\mathbf{a})$*: if Y, then agent $\mathbf{a}$ desires $\sim s$, which expresses that it does not like to be sanctioned.*
7. $able(\mathbf{n}, V(n, \mathbf{a}), Y \cup \{\sim x\})$*: system $\mathbf{n}$ is able to achieve $V(n, \mathbf{a})$ in context Y and $\sim x$;*
8. $able(\mathbf{n}, s, Y \cup \{V(n, \mathbf{a}), \sim x\})$*: system $\mathbf{n}$ is able to achieve s in context Y, $V(n, \mathbf{a})$ and $\sim x$.*

An obligation $O_\mathbf{an}(x, s|Y)$ is an ought-to-do obligation when $x \subseteq L(X_\mathbf{a})$, and an ought-to-be obligation otherwise.

The following example illustrates that the agent does not always know how to fulfill ought-to-be obligations.

Example 2. Consider $\langle A, X, D, G, AD, E, MD, \geq, \mathbf{n}, N, V, GD \rangle$ with $A = \{\mathbf{a}, \mathbf{n}\}$, $X_\mathbf{a} = \{b\}$, $X_\mathbf{n} = \{V(n, \mathbf{a}), m\}$, $P = \{p, x\}$, $N = \{n\}$, $MD(GD(\mathbf{a})) = \{\top \rightarrow x\}$. Agent $\mathbf{a}$ desires $\neg s$: $MD(D_\mathbf{a}) = \{\top \rightarrow \neg s\}$. Agent $\mathbf{n}$ desires and goals are: $MD(D_\mathbf{n}) = \{\top \rightarrow x, \neg x \rightarrow V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow s, \top \rightarrow \neg V(n, \mathbf{a}), \top \rightarrow \neg s\}$ and $MD(G_\mathbf{n}) = \{\top \rightarrow x, \neg x \rightarrow V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow s, \top \rightarrow \neg V(n, \mathbf{a}), \top \rightarrow \neg s\}$.

Assume $E = \{\top \rightarrow p, b \rightarrow x, p \rightarrow \neg x\}$. In this situation, agent $\mathbf{a}$ cannot fulfill the obligation, because there does not exist a decision $\delta = \delta_\mathbf{a} \cup \delta_\mathbf{n}$ such that $x \in outE(E, \delta)$. The parameter x could be achieved by means of action b since $b \rightarrow x \in E$. But in the initial state the parameter p is true ($\top \rightarrow p \in E$) and $p \rightarrow \neg x \in E$: hence, $\{b\}$ is not a decision of agent $\mathbf{a}$, otherwise we have $\{x, \neg x\} \in outE(E, \delta')$.

5 Formalization of Violation Games of Attacker

The basic picture is visualized in Figure 1 and reflects the deliberation of agent **a** in various stages. This figure should be read as follow. Agent **a** is the decision maker: it is making a decision $\delta_{\mathbf{a}}$, and it is considering the effects of the fulfilment or the violation of the obligations it is subject to. To evaluate a decision $\delta_{\mathbf{a}}$ according to its desires and goals ($D_{\mathbf{a}}$ and $G_{\mathbf{a}}$), it must consider not only its actions, but also the reaction of system **n**: **n** is the normative system, which may recognize and sanction violations. Agent **a** recursively models system **n**'s decision $\delta_{\mathbf{n}}$ (that system **n** takes according to agent **a**'s point of view), typically whether it counts the decision $\delta_{\mathbf{a}}$ as a violation and whether it sanctions agent **a** or not, and then bases its decision on it. Now, to find out which decision system **n** will make, agent **a** has a *profile* of system **n**: it has a representation of system **n**'s motivational state. When agent **a** makes its decision and predict system **n**'s decision, we assume in this paper that it believes that system **n** is aware of it.

Definition 7 (Recursive modelling). *Let*

$$NMAS_2 = \langle A, X, D, G, AD, E, MD, \geq, \mathbf{n}, N, V, GD \rangle$$

be a normative multiagent system.

- *Let the unfulfilled motivations of decision δ for agent $a \in A$ be the set of motivations whose body is part of the closure of the decision under the effect rules but whose head is not.*
  $U(\delta, a) = \{m \in M \cap MD(a) \mid MD(m) = l_1 \wedge \ldots \wedge l_n \rightarrow l, \{l_1, \ldots, l_n\} \subseteq outE(E, \delta)$ *and* $l \notin outE(E, \delta)\}$
- *A decision δ (where $\delta = \delta_{\mathbf{a}} \cup \delta_{\mathbf{n}}$) is* optimal *for agent* **n** *if and only if there is no decision $\delta'_{\mathbf{n}}$ such that $U(\delta, \mathbf{n}) >_{\mathbf{n}} U(\delta_{\mathbf{a}} \cup \delta'_{\mathbf{n}}, \mathbf{n})$. A decision δ is optimal for agent* **a** *and agent* **n** *if and only if it is optimal for agent* **n** *and there is no decision $\delta'_{\mathbf{a}}$ such that for all decisions $\delta' = \delta'_{\mathbf{a}} \cup \delta'_{\mathbf{n}}$ and $\delta_{\mathbf{a}} \cup \delta''_{\mathbf{n}}$ optimal for agent* **n** *we have that $U(\delta', \mathbf{a}) >_{\mathbf{a}} U(\delta_{\mathbf{a}} \cup \delta''_{\mathbf{n}}, \mathbf{a})$.*

In Example 3, the desire of agent **a** (that p is true) conflicts in an indirect way with x, the normative goal of $O_{\mathbf{an}}(x, s \mid \top)$. The effect rules E represent the incompatibility of the effects of the two decision variables x (which has effect $\neg p$) and b (which achieves p). $\{x, b\}$ is not a decision of agent **a** since $outE(E, \{x, b\}) = \{x, b, p, \neg p\}$ is not consistent.

The choice between x and b is taken by comparing the results of recursive modelling: if x then $outE(E, \delta_{\mathbf{a}} \cup \delta_{\mathbf{n}}) = \{x, \neg b, \neg p, \neg V(n, \mathbf{a}), \neg s\}$ and if b then $outE(E, \delta_{\mathbf{a}} \cup \delta_{\mathbf{n}}) = \{b, \neg x, V(n, \mathbf{a}), s\}$. Since agent **a** prefers not being sanctioned with respect to leaving p unfulfilled, it chooses x. The priority order on the motivations is implicitly represented by the numerical index of the rules. A higher number represents a higher priority.

Example 3. $O_{\mathbf{an}}(x, s \mid \top)$

		P p	
		E $x \rightarrow \neg p, b \rightarrow p$	
Agent **a**		Agent **n**	
$X_{\mathbf{a}}$	x, b	$X_{\mathbf{n}}$	$V(n, \mathbf{a}), s$
$D_{\mathbf{a}}$	$\top \rightarrow^2 \neg s, \top \rightarrow^1 p$	$D_{\mathbf{n}}$	$\top \rightarrow^5 x, \neg x \rightarrow^4 V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow^3 s,$ $\top \rightarrow^2 \neg V(n, \mathbf{a}), \top \rightarrow^1 \neg s$
$G_{\mathbf{a}}$		$G_{\mathbf{n}}$	$\top \rightarrow^5 x, \neg x \rightarrow^4 V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow^3 s$
$\delta_{\mathbf{a}}$	$x, \neg b$	$\delta_{\mathbf{n}}$	$\neg V(n, \mathbf{a}), \neg s$
$U_{\mathbf{a}}$	$\top \rightarrow^1 p$	$U_{\mathbf{n}}$	
	$outE(E, \delta)$	$x, \neg b, \neg p, \neg V(n, \mathbf{a}), \neg s$	

In the next example, agent **a** has to choose between two obligations which, even if they are not explicitly conflicting, it cannot respect at the same time, since fulfilling the first one (the *ought-to-do* obligation $O_{\mathbf{an}}(x, s \mid \top)$) makes it impossible to do anything for the second one (the *ought-to-be* obligation $O_{\mathbf{an}}(p, s' \mid \top)$).

Example 4. $O_{\mathbf{an}}(x, s \mid \top), O_{\mathbf{an}}(p, s' \mid \top)$

		P p	
		E $x \rightarrow \neg p, y \rightarrow p$	
Agent **a**		Agent **n**	
$X_{\mathbf{a}}$	x, y	$X_{\mathbf{n}}$	$V(n, \mathbf{a}), s, V(n', \mathbf{a}), s'$
$D_{\mathbf{a}}$	$\top \rightarrow^2 \neg s, \top \rightarrow^1 \neg s'$	$D_{\mathbf{n}}$	$\top \rightarrow^{10} x, \neg x \rightarrow^9 V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow^8 s,$ $\top \rightarrow^4 \neg V(n, \mathbf{a}), \top \rightarrow^3 \neg s,$ $\top \rightarrow^7 p, \neg p \rightarrow^6 V(n', \mathbf{a}), V(n', \mathbf{a}) \rightarrow^5 s',$ $\top \rightarrow^2 \neg V(n', \mathbf{a}), \top \rightarrow^1 \neg s'$
$G_{\mathbf{a}}$		$G_{\mathbf{n}}$	$\top \rightarrow^{10} x, \neg x \rightarrow^9 V(n, \mathbf{a}), V(n, \mathbf{a}) \rightarrow^8 s,$ $\top \rightarrow^7 p, \neg p \rightarrow^6 V(n', \mathbf{a}), V(n', \mathbf{a}) \rightarrow^5 s'$
$\delta_{\mathbf{a}}$	$x, \neg y$	$\delta_{\mathbf{n}}$	$\neg V(n, \mathbf{a}), s, V(n', \mathbf{a}), s'$
$U_{\mathbf{a}}$	$\top \rightarrow^1 \neg s'$	$U_{\mathbf{n}}$	$\top \rightarrow^7 p, \top \rightarrow^2 \neg V(n', \mathbf{a}), \top \rightarrow^1 \neg s'$
	$outE(E, \delta)$	$x, \neg y, \neg p, \neg V(n, \mathbf{a}), \neg s, V(n', \mathbf{a}), s'$	

In the next example, we again model the sanction s as a parameter which is made true by a decision variable $m \in X_{\mathbf{n}}$. However, this time agent **a** does not directly make the sanction impossible. Rather, it triggers some goals of agent **n**. We examine how agent **a** exploits the recursive modelling to influence the behavior of the normative agent. Besides the usual goals and desires described by the obligation to do x, here we assume that, in a situation where p is true, agent **n** has the goal to make the decision variable $r \in X_{\mathbf{n}}$ true. So, given $p \in P$, it would like to choose decision $\delta_{\mathbf{n}} = \{V(n, \mathbf{a}), m, r\}$: but the two decision variables m and r are incompatible. Since the conditional desire $p \rightarrow r \in out(D_{\mathbf{n}})$ is preferred to $V(n, \mathbf{a}) \rightarrow s \in out(D_{\mathbf{n}})$, agent **n** recognizes the violation ($V(n, \mathbf{a})$) but it does not sanction agent **a**.

Example 5. $O_{\mathbf{an}}(x, s \mid \top)$

P	p, s
E	$b \to p, m \to s, r \to \neg s$
Agent **a**	Agent **n**
$X_{\mathbf{a}}$ x, b	$X_{\mathbf{n}}$ $V(n, \mathbf{a}), m, r$
$D_{\mathbf{a}}$ $\top \to^3 \neg s, \top \to^2 \neg x, \top \to^1 \neg b$	$D_{\mathbf{n}}$ $p \to^6 r, \top \to^5 x, \neg x \to^4 V(n, \mathbf{a}),$ $V(n, \mathbf{a}) \to^3 s,$ $\top \to^2 \neg V(n, \mathbf{a}), \top \to^1 \neg s$
$G_{\mathbf{a}}$	$G_{\mathbf{n}}$ $p \to^6 r, \top \to^5 x, \neg x \to^4 V(n, \mathbf{a}),$ $V(n, \mathbf{a}) \to^3 s$
$\delta_{\mathbf{a}}$ $b, \neg x$	$\delta_{\mathbf{n}}$ $V(n, \mathbf{a}), r, \neg m$
$U_{\mathbf{a}}$ $\top \to^1 \neg b$	$U_{\mathbf{n}}$ $\top \to^5 x, V(n, \mathbf{a}) \to^3 s, \top \to^2 \neg V(n, \mathbf{a})$
$outE(E, \delta)$	$\neg x, p, V(n, \mathbf{a}), b, \neg m$

6 Summary

With the increase of multiagent systems with explicit norms, their security becomes an urgent problem. To deal with a wide range of possible attacks, we need an expressive attacker model. In this paper we considered the abilities or power of agents, and two new kinds of examples, either due to the inability or lack of power of the attacker itself, or due to the inabilities or lack of power of the normative system. A topic of further research is the extension of our model to reason about beliefs and observations of attackers and normative systems [1].

References

1. Boella, G., van der Torre, L.: Fulfilling or violating obligations in normative multiagent systems. In: Procs. of IAT'04, pp. 483–486. IEEE, Los Alamitos (2004)
2. Boella, G., van der Torre, L.: Normative multiagent systems and trust dynamics. In: Falcone, R., Barber, S., Sabater-Mir, J., Singh, M.P. (eds.) Trusting Agents for Trusting Electronic Societies. LNCS (LNAI), vol. 3577, pp. 1–17. Springer, Heidelberg (2005)
3. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. Computation and Mathematical Organizational Theory, Special issue on Normative Multiagent Systems 12(2-3), 71–79 (2006)
4. Boutilier, C.: Toward a logic for qualitative decision theory. In: Procs. of KR-94, Bonn, pp. 75–86 (1994)
5. Broersen, J., Dastani, M., Hulstijn, J., van der Torre, L.: Goal generation in the BOID architecture. Cognitive Science Quarterly 2(3-4), 428–447 (2002)
6. Gmytrasiewicz, P.J., Durfee, E.H.: Formalization of recursive modeling. In: Procs. of ICMAS'95, pp. 125–132. AAAI/MIT Press, Cambridge, MA (1995)
7. Lang, J., van der Torre, L., Weydert, E.: Utilitarian desires. Autonomous Agents and Multiagent Systems 5(3), 329–363 (2002)
8. Makinson, D., van der Torre, L.: Input-output logics. Journal of Philosophical Logic 29(4), 383–408 (2000)

An Environment to Support Multi-Party Communications in Multi-Agent Systems

Julien Saunier[1] and Flavien Balbo[1,2]

[1] LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, Paris Cedex 16
[2] GRETIA, INRETS, 2, Avenue du Général Malleret-Joinville, -F94114 Arcueil
{balbo,saunier}@lamsade.dauphine.fr

Abstract. Two-party communication is the most-studied model to support interaction between two cognitive agents, whereas that is only one case of what an agent should be able to do. Multi-party communications enhance this model, by taking into account all the roles an agent can have in a communication. Nevertheless, there are no generic models and infrastructures that enable to apply multi-party communication in a standardized way. We emphasize that the environment, in the sense of a common medium for the agents, is a suitable paradigm to support multi-party communication. We propose a general and operational model called Environment as Active Support of Interaction (EASI), that enables each agent to actively modify the environment according to its communication needs. Algorithms are proposed and assessed with an example stemming from the ambient intelligence domain.

1 Introduction

Traditionally in Multi-Agent Systems (MAS), communications involve two interlocutors, the sender that initiates the interaction and the addressee. In such dialogues, the interlocutors know each-other but this bilateral communication model is only one case of what agents should be able to do. An agent should be able to communicate with groups of agents without broadcast or parallel dialogues. Moreover, an agent should be able to choose the receiver(s) of its messages thanks to other information than their address. Multi-party communication (MPC) [2,4,6,7] is an answer to these new expectations: several agents can hear the same message, and they may have different roles in the communication.

The design of an architecture for MPC has to take into account several interaction models. When a message is exchanged, the interaction is direct if the sender knows explicitly the addressee(s), and the interaction is indirect/mediated if the sender does not choose an addressee. Context awareness can be necessary to the interaction process if the sender chooses explicitly only a part of the addressees or if it chooses them thanks to other information than the address of the agents. In this paper, an infrastructure that enables multi-party communication in a standardized way is proposed. Based on a communication environment modeling, our proposition called Environment as Active Support of Interaction (EASI) enables both direct and indirect interaction, in the context of reliable networks.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 52–61, 2007.

This paper is organized as follows. Section 2 describes the issues of multi-party communication and a motivating example of ambient intelligence application is given. Section 3 introduces the EASI description model. Section 4 gives an algorithm that enables to find the receivers of a communication. Section 5 evaluates the validity of this algorithm on our example and section 6 concludes.

2 Multi-Party Communications

In a MPC, all the agents receiving a message are called *recipients*. This generic name hides the different roles that the agents can have in a communication act. For example, addressing a warning to a group of students can be heard by passers-by. In this case the students and the passers-by are recipients but do not have the same role in the communication act, and this difference affects their reactions. There are three criteria to qualify the role of an agent in a communication. The first is the *intention* criterion: Is the agent expected to take part in the communication, and if so what is its intended role? If a recipient is intended, it may be expected to take an active part in the conversation, or just to hear passively what is exchanged. The second is the *knowledge* criterion: Is the agent known by its interlocutors as taking part in the communication? The sender can be anonymous and does not necessarily know *a priori* who will overhear its message, nor their address, e.g. a public announcement to "every art student". The third is the *initiative* criterion: Is the communication perception the result of an action of the sender, or of the recipient? Hearing a communication may be the result of an initiative of the sender, for example a multicast decided by the sender, or of an initiative of the recipient, for example agents reading voluntarily a forum. According to these criteria, the different roles that an agent can play in a communication are defined in Fig. 1. For example, if the sender and the recipient share a common knowledge about the communication act, then the recipient is either an *auditor* or an *addressee*. If a recipient has not been chosen by the sender then this recipient is either an *overhearer* or an *eavesdropper*.

For the same communication act, the recipient agents have one of these roles and the difficulty is to find all the recipients of this message. There are two problems: 1) the sender does not choose all the recipients; 2) the sender does not know explicitly all the recipients. The sender does not compulsorily choose all the recipients since by definition an overhearer receives a message according to its own initiative. That means that the sender does not control the communication channel, like for instance in a wifi

Role	Intention	Knowledge	Initiative
Sender	Intended or not	Known or not	Sender
Addressee	Intended, Active	Known	Sender
Auditor	Intended, Passive	Known	Sender
Reception Group	Intended, Active or Passive	Members are not individually known	Sender
Overhearer	Unintended	Known or not	Overhearer
Eavesdropper	Unintended	Unknown	Eavesdropper

Fig. 1. Roles of the agents in a multi-party communication

communication. For classical networks, this has to be achieved through the use of a middleware. The sender does not compulsorily know the specific recipients of its message but it must be able to express constraints on them [7]. These constraints can be expressed by conditions on properties of the environment of the agent (other agents, objects, etc.). Indeed, MPC implies to be able to communicate thanks to other criteria than the address of the agents. An agent should be able to contact, for example, "the second-year students in the gymnasium". That means that a multi-party middleware has to support context awareness in the sense that the choice of the agents involved in the communication act is done thanks to the evaluation of the context.

Although context awareness has long been considered as an undergone state, it has emerged recently that awareness is an active state, and not only the result of stimuli. Works in the fields of psychology and sociology have discussed whether or not there also has to be an active participation of the "perceiver". For example, Heath [5] shows that awareness is not only a state of availability to the environment, but that it is also an ability to "filter relevant information which is of particular significance". Awareness is directed towards a subject via a receptor, it is a receptive activity that is the result of decisions of action. Functionally, this means that stimuli are conveyed by the environment. Context awareness is generally studied from the point of view of the perceiver. However, directing a message in the sense of choosing the recipients according to properties is a particular kind of focus, which denotes the initiative of the sender.

To sum up, a middleware for MPC has to take into account a multiplicity of roles in the communication process. For an agent, the choice of a role may be the result of its own initiative and/or be conditioned by the context. This implies that the sender is not the only one that controls the communication process and that the agents can adapt independently the middleware according to their needs. Moreover, the context-awareness is related to an active availability to the environment. Hence, our proposition is to design an active communication environment, which mediates the interaction.

Illustrative Example: This paper shows a communication service that has been designed as an example of an Ambient Intelligence application (AmI). This application supports the interaction between the company employees and their visitors. Each of them is represented by an agent that manages its communications. An *employee* agent belongs to a department, and has an availability. The company building is made up of department offices, meeting rooms and a reception. The objective here is to propose an application that will support in a standardized way all the different interaction needs. There are direct interactions, for example, a visitor asks for a particular employee (situation noted S_{dir}). There are indirect interactions, for example, the release of a meeting room is an event that has to be perceived by related agents (S_{ind}). The application must also support more complex interaction models: the sender should be able to use context awareness to choose its addressee, for example a visitor asks an available employee from a given department (S_{awa}), and the agents should be able to use context awareness to choose their messages, for example, an agent can monitor the activity by listening to the employees in a particular department meeting room with a visitor (S_{mon}).

3 Environment as Active Support of Interaction

With EASI, direct/indirect communication processing or event/data processing are possible using the same medium, the environment. In the following, the action of finding the good recipients for a sender and/or the good sender for a recipient is called a $connection$. In our proposition, the environment is a communication middleware that contains all the required information about the components of the MAS (agents, messages,...) that are clustered in order to find the correct components for each connection efficiently. That is why the EASI formalism is based on the Symbolic Data Analysis (SDA) theory [1], that is a clustering model for large qualitative and quantitative data sets. In EASI, modeling the MAS gives the interaction information clustering and the SDA formalization enables their reification.

Definition 1 (Environment). *$E = \langle \Omega, P, \mathcal{F} \rangle$ is the communication environment, with:*

- *$\Omega = \{\omega_1, ..., \omega_m\}$ the set of m entities, with $A \subset \Omega$ the set of agents.*
- *$P = \{p_1, ..., p_n\}$ the set of n observable properties.*
- *$\mathcal{F} = \{f_1, ..., f_k\}$ the set of k filters.*

The first component of our environment model is Ω, the set of entities. An entity $\omega_l \in \Omega$ is the description by observable properties of a component of the MAS (agent, percept, object). An observable property $p_i \in P$ is a function that gives for an entity ω_l a value that can be used for a connection, $\forall p_i \in P,\ p_i : \Omega \rightarrow d_i \cup \{unknown, null\}$, with d_i the description domain of p_i. d_i can be quantitative, qualitative or a finite data set. The $unknown$ value expresses that the property exists for the entity but the value has not been given. The value of a property can be modified dynamically at run-time, except if it is $null$. A $null$ value expresses the absence of a property, and this cannot be changed. We call a $Pdescription$ the set of properties that describes an entity (noted $P_\omega = \{p_i \in P | p_i(\omega) \neq null\}$ for the entity ω) or a set of entities (noted $P_S = \{p_i \in P | \forall \omega \in S, p_i(\omega) \neq null\}$ for the set of entities S). The properties used to describe the entities are not set *a priori*, which means that the agents must share some meta-knowledge about the components of the MAS.

The last component of our environment model is $\mathcal{F}$ the set of filters, each filter being the description of the constraints on the observable properties of the entities that are related to a connection. Let e be an elementary test, $\omega_l \in \Omega, p_i \in P$, $e : \Omega \rightarrow \{true, false, unknown\}, e(\omega_l) = [P_i(\omega_l)\ R\ v]$ with R a comparison operator and $v \in d_i \cup \{unknown\}$. The value of e is $true$ if the test holds, $false$ if not and $unknown$ if the value depends on a variable (the value is given by a matching process). For example, $\omega_l \in \Omega, e_{position}(\omega_l) = [position(\omega_l) \in MR]$ (with MR the set of meeting rooms) is a test related to the $position$ of an entity. For an agent ω_j, the test is true if ω_j is in a meeting room, false if not. The $false$ value includes the cases where the value of the property is $unknown$. The tests are done on entities having the required properties, hence it is not possible for a property to match a $null$ value. In SDA, a symbolic object is a coherent description of entities. An assertion is a special case of a symbolic object and is a conjunction of elementary tests: $\forall \omega_l \in \Omega\ as(\omega_l) = \wedge_{i=1,...,n}[p_i(\omega_l) R_i d_i]$. The extent of an assertion in Ω contains all the entities that satisfy its description. For example, the assertion $as_1(\omega_l) = e_{pos}(\omega_l) \wedge [dep(\omega_l) = \text{“}marketing\text{”}]$ describes the

entities that belong to the "marketing" department and are located in a meeting room. Remember that a filter is a reification of a *connection*, this is why it is a symbolic object which describes the entities that are related to a connection.

Definition 2 (Filter). *A filter $f \in \mathcal{F}$ is a tuple $\langle f_{ag}, f_{pe}, [f_{co}], n_f \rangle$ where:*

- *f_{ag} is the description of the recipient agent, such that:* $a \in A, f_{ag}(a) = \wedge_{p_i \in P_{f_{ag}}} [p_i(a) R_{p_i}^{f_{ag}} d_{p_i}^{f_{ag}}]$.
- *f_{pe} is the description of the percept, such that:* $\omega \in \Omega, f_{pe}(\omega) = \wedge_{p_i \in P_{f_{pe}}} [p_i(\omega) R_{p_i}^{f_{pe}} d_{p_i}^{f_{pe}}]$.
- *f_{co} is the optional description of the context, such that:* $C \subset \Omega, f_{co}(C) = \wedge_{\forall c \in C} f_{co}(c)$, *with* $f_{co}(c) = \wedge_{p_i \in P_c} [p_i(c)\ R_{p_i}^c\ d_{p_i}^c]$.
- *n_f is the name of the filter.*

The recipient Pdescription $P_{f_{ag}}$ (definition 2) gives the properties that an agent must have to be a potential recipient. Using $R^{f_{ag}}$, which is a tuple of comparison operators, and $d^{f_{ag}}$, which is a tuple of reference values and variables, the assertion f_{ag} describes the conditions to become a receiver. In the same way, the description of the percept is given by the assertion f_{pe}. The context of the connection is given by the symbolic object f_{co}. From this viewpoint, a context is a subset of Ω and therefore a part of the observable state of the MAS. In order to describe an interaction, the first two assertions are compulsory. The extent of a filter according to the property existence constraint in Ω contains the potential components of a connection that are gathered in the tuple $\langle E(P_{f_{ag}}), E(P_{f_{pe}}), E(P_{f_{co}}) \rangle$, with for example $E(P_{f_{ag}}) = \{a \in A | \forall p_i \in P_{f_{ag}}, p_i(a) \neq null\}$.

Figure 2 shows a simple instantiation of our environment modeling for AmI. Here there are four entities, $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ that are respectively the description of the agent *visitor* v_1, of the agents *employee* e_1 and e_2 and of the message m_1. There is a property called *pos* (for *position*) and d_{pos} contains the rooms in the building. The

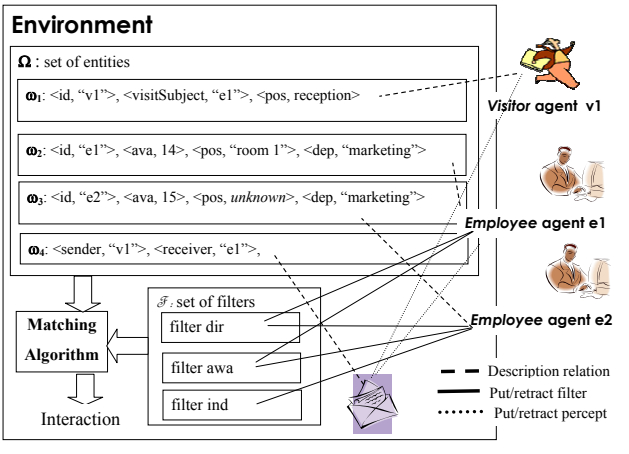


Fig. 2. EASI interaction modeling example

value of $pos(\omega_1)$ is a room of the building; the value $pos(\omega_2)$ is $unknown$ because the value has not been given; the value $pos(\omega_4)$ is $null$ because ω_4 does not have this property in its description. There are three filters, $\mathcal{F} = \{dir, awa, ind\}$. An agent puts in the environment the filter that describes the components of the connection related to its needs. For example, the *employee* agents have two common filters $\{dir, awa\}$ and the *employee* agent $e2$ has put the filter ind. The *visitor* agent v_1 puts in the environment the description ω_4 of the message m_1 (the message itself is stored in another part of the environment). Thanks to the description of the components of the MAS (Ω), the matching process described below dispatches m_1 thanks to $\mathcal{F}$. An example of filter is $a \in A, io \in \Omega, f_{S_{awa}} = \langle [dep(a) = ?x] \wedge [ava(a) < t], [dep(io) = ?x], \emptyset, \text{“}awa''\rangle$ (with t the current time value and $?x$ a variable). This filter describes the condition related to the agents ($[dep(a) = ?x] \wedge [available(a) < t]$), and to the percept ($[dep(io) = ?x]$).

4 Matching Process

The difficulty in the MPC processing is that each percept could be interesting for each agent according to the context of the MAS. With EASI this complexity is reduced to the set of entities related to a connection. Our first proposition is to limit the validation to the entities that have the required properties. The extent of a filter f is the tuple of sets $\langle E(P_{f_{ag}}), E(P_{f_{pe}}), E(P_{f_{co}})\rangle$. These sets are computable and are not updated for the same MAS description. A percept may be received by several agents using the same filter. For example, for the same percept, $f_{S_{awa}}$ is valid for all the agents that are available in the required department. Moreover, a percept may be received by several agents using several filters. For example, for the same percept, $f_{S_{dir}}$ and $f_{S_{mon}}$ may be valid for several agents. The difficulty is to find all the potential recipients using the set of filters that are valid for a percept.

Definition 1 (Clustering of the components of a connection). *Let $f \in \mathcal{F}$ be a filter, $io \in \Omega$ be a percept, $a \in A$ be an agent, and $C \subset \Omega$ be a context:*

- $Cha_{io} = \{f \in \mathcal{F} | io \in E(P_{f_{pe}})\}$ *is the set of filters related to the percept io,*
- $Rec_{io} = \{a \in A | \exists f \in Cha_{io}, a \in E(P_{f_{ag}})\}$ *is the set of the potential receivers of the percept io,*
- $Co_{io} = \{C \subset \Omega | \exists f \in Cha_{io}, C \in E(P_{f_{co}})\}$ *is the set of the potential contexts of the percept io,*
- $FPer_a = \{f \in \mathcal{F} | a \in E(P_{f_{ag}})\}$ *is the set of filters related to the agent a,*
- $FCo_C = \{f \in \mathcal{F} | C \in E(P_{f_{co}})\}$ *is the set of filters related to a context C.*

With the sets defined in Def. 1, for each percept the other components that are related to it are known. For example, the set Rec_{io} contains all the agents that have the required properties to be taken into account as potential recipient of the percept io.

In Algo 1, the matching is based on the following relation, $V : A \times \Omega \times P(\Omega) \times \mathcal{F} \rightarrow \{true, false\}$. A filter $f \in \mathcal{F}$ is valid $\Leftrightarrow a \in A, io \in \Omega, C \subset \Omega, V(a, io, C, f) = f_{ag}(a) \wedge f_{pe}(io) \wedge f_{co}(C)$. An advantage with EASI is that the recipient knows the context in which it receives the percept. Let PK_a be what we call the Private Knowledge of the agent a, when $V(a, io, C, f)$ is valid, then $receive(a, f)$ is triggered with

Algorithm 1. The *structural* clustering based matching algorithm

$io \in \Omega$ a new percept
ForEach ($a \in Rec_{io}$)
ForEach ($C \in Co_{io}$)
ForEach ($f \in (FPer_a \cap Cha_{io} \cap FCo_C)$)
If ($V(a, io, C, f)$) then $receive(a, f)$ EndIf
EndForEach
EndForEach
EndForEach

$receive(a, f) \Leftrightarrow PK_a \leftarrow PK_a \cup \{io, C', n_f\}$ and $C' \subset C$. The agent receives the percept, a part of the context (in fact a copy) and the name of the filter.

Each initiative of the agents is represented by a filter, whatever their role in the communication is (Fig. 1). Hence, once a percept has been matched with all the filters, all the initiatives are taken into account, which enables EASI to be generic towards the interaction models, in a standardized way. The distinction relies on the agent, called *initiator* agent, that puts the filter in the environment. Let f be a filter: if the *initiator* agent of f does not belong to $E(f_{ag}) = \{a \in A | f_{ag}(a) = true\}$ then f_{ag} describes the agent(s) with which the *initiator* agent wants to interact, which is a direct interaction. In AmI, the example of direct interaction is $f_{S_{dir}} = \langle [id(a) =?x], [receiver(io) =?x], \text{“}dir''\rangle$. If the *initiator* agent belongs to $E(f_{pe})$ then f_{pe} describes the percept that it wants to receive, which is an indirect interaction. In AmI, the example is $f_{S_{ind}} = \langle [id(a) = \text{“}e2''], [pos(io) \in MR] \wedge [sub(io) = \text{“}available''], \text{“}ind''\rangle$. The agent with the id value “e2” will receive all the percepts related to the availability of the meeting rooms. Moreover, this algorithm takes into account more complex interaction models such as awareness. The *initiator* agent belongs to $E(f_{ag})$ but the percept is addressed to another agent. In AmI, the example is $f_{S_{mon}} = \langle [id(a) = \text{"}e3\text{"}], [sender(io) =?y], [pos(ax) \in MR] \wedge [pos(ax) =?x] \wedge [pos(ay) =?x] \wedge [dep(ax) = \text{"}sav\text{"}] \wedge [id(ax) =?y], \text{“}mon''\rangle$ with $ax, ay \in A$.

If the observable properties are not often updated, it is possible to anticipate that among the potential receiver of a percept a subset of them does not satisfy the required condition. In the preceding algorithm they will be evaluated in spite of this knowledge. The ideal algorithm respects the processing of the preceding algorithm but is based on the extent $E(f_{ag})$, $E(f_{pe})$ and $E(f_{co})$ in Ω. For example, that means that $Cha_{io} = \{f \in \mathcal{F} | io \in E(f_{pe})\}$ and that this set as well as the other sets contains exactly the entities that match the constraints. Because the filters may require matching between entities of its different composing set, these extents are rarely computable.

We propose to change in the Algo 1 the matching sets by sets without the entities having a value that does not satisfy the filter. For example, the restricted set of potential receivers is $\forall io \in \Omega, Rec^v_{io} = \{a \in Rec_{io} | \forall p_i \in P_{f_{ag}}, [p_i(a) R_{p_i}^{f_{ag}} d_{p_i}^{f_{ag}}] \neq false\}$. Following the same principle Co^v_{io} is the set of contexts that can satisfy the filter and for the agent a, $Fper^v_a$ is the subset of $FPer_a$ that the agent may satisfy. This computation may be done for all the sets related to the preceding algorithm, each of them is noted with the power v for *valid* (Algo 2). The result is a more efficient processing of

Algorithm 2. The *conditional* matching algorithm

$io \in \Omega$ a new percept
 ForEach ($a \in Rec_{io}^{v}$)
  ForEach ($C \in Co_{io}^{v}$)
   ForEach ($f \in (FPer_{a}^{v} \cap Cha_{io} \cap FCo_{C}^{v})$
    If ($V(a, io, C, f)$) then $receive(a, f)$ EndIf
   EndForEach
  EndForEach
 EndForEach

the communication but the cost is the maintenance of the sets. An entity that is "valid" for a filter may be updated and become "invalid", the reverse may also be true.

5 Experiments and Empirical Results

To evaluate the performance of our algorithms, we have set up a series of comparative tests including the broadcast that is a classic way to support MPC and our algorithms 1 and 2, noted EASI1 and EASI2. In particular, we are interested in the dependency between the update rate, the number of agents and the performance of the system. The tests are simulations of the AmI example described throughout the paper.

Settings. With a broadcast, the connection problem solving is decentralized, i.e. each agent computes the receivers of its message, whereas EASI centralizes this computation. To compare the two approaches, we have to evaluate the performance of the system as a whole. Hence both the actions of the agents and the processing of our environment are measured in a centralized simulator. The trade-off is that we do not take into account the network bandwidth costs.

For each time step, the agents are executed in random order. An agent checks its messages, and then chooses a behavior: answering a message, adding a filter, etc. Half of the agents are *employee* agents, and the other half *visitor* agents. Every agent updates its property $available$ when necessary, and the *employee* agents modify their $position$ property according to an update rate. In order to create the conditions for the indirect messages to be received, the number of rooms is 20% below needs.

Two kinds of information can be broadcast: messages and updates. In the first case, the receivers decide what interests them, but they cannot know the properties of the other entities directly. For instance $f_{S_{mon}}$ cannot be implemented as such because the receiver has to know the current values of the properties of a_x and a_y. In the second case, the senders choose the receivers, but the senders have to know what interests the receivers. We have chosen the second solution which uses the same criteria as the EASI algorithms.

Results. The stability[1] of the runs is good. The first chart (Fig. 3) gives the run-time of the simulation as a function of the number of agents. The broadcast is the least efficient at each time step, the distance from the other curves increasing at a non-linear rate.

[1] The standard deviation on 100 runs is below 0.08% of the average run time.

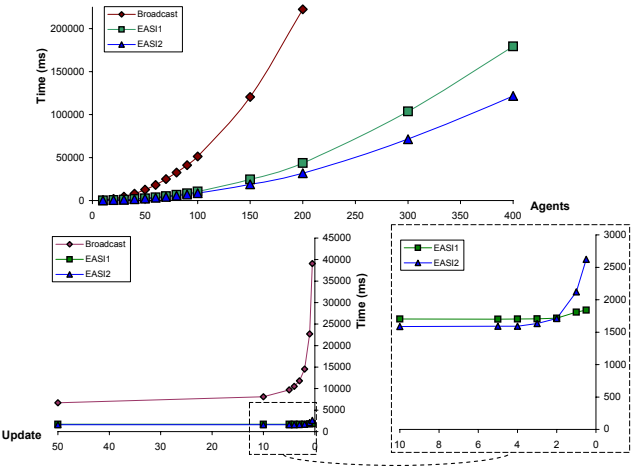


Fig. 3. Broadcast vs EASI Algorithm 2 - EASI Algorithm 3. Run time (in milliseconds) as a function of the number of agents (top) and the update frequency (bottom), average on 30 runs.

It is the classical limit of broadcast which means it cannot be used when the number of messages and/or agents is high. In order to test our proposition with a fairly large number of agents, when broadcast is impossible, we have run tests for up to 1000 agents and 6000 steps in 9 minutes, which represents 22 million messages.

For less than 30 agents, run-time for EASI2 is longer than for EASI1, while for more agents it is the contrary. The reason is that the creation and update management of the sets used by EASI2 take more time than for the sets used by EASI1, whereas the processing of a message is faster. Hence, whereas for a low number of agents the additional cost per set are not offset by the processing, EASI2's advantage increases with the number of agents.

The second chart displays the run-time of the simulation as a function of the update rate. Once again, the broadcast is clearly at a disadvantage compared with our algorithms. Since EASI1 uses sets based on the static clustering of the entities, the update rate has no effect on the algorithm itself, the small increase in run-time being due to the agents' computations. Finally, EASI2's run-time is sensitive to high dynamicity (third chart): when the update rate is below 1/2, it is faster than EASI1, but over 1/2 it becomes less efficient because the cost of updating the sets becomes too high.

These tests show that the EASI model and our algorithms are a valid solution to the connection problem. The choice between EASI1 and EASI2 has to be made according to the size and the dynamicity of the MAS.

6 Conclusion

Most of the work on multi-party communication are related to the difference between two-party and multi-party dialogues [2]. In [4], Dignum and Vreeswijk describe the roles that an agent can have in a MPC and discuss the consequences for a negotiation

process. On the same way, in [7] the authors argues for an extension of the semantics of the agent communication language to take into account the group communication specificities. These works are complementary to our own, there is a need for both specific semantics and a middleware. From a practical viewpoint, there is no specific middleware related to MPC. The most usual proposition is the use of a dedicated blackboard [4,6] that is a static medium for the sharing of the messages. The tuple-space model, which was initiated by Linda [3], comes within this category even though it does not specifically address multi-agent systems. The difference with our work is that in Linda-like models agents (processes) are not represented by data inside the environment, while in EASI they are. The consequence is that the interaction between two agents can be conditioned by their state. In addition, our use of symbolic objects allows for a richer expressiveness than the templates used in these models. Remember that a template tests whether the data considered matches the template, whereas with a symbolic object, the matching is conditioned by a context, i.e. an environmental criterion.

We have introduced EASI and drawn its main features, and especially the flexible management of the MAS communications. In this proposal, an agent states its *initiatives* declaratively and the connection is established by the environment. The use of symbolic objects enables to take into account in a standardized way the needs of the agents and the context of the communications. Hence, EASI allows multi-party communications, and the experiments show the feasibility of this environment model. Future work will focus on the addition of controls over agent communications using norms and laws. We are also investigating applications such as crisis management simulations.

References

1. Bock, H., Diday, E.: Analysis of symbolic data. exploratory methods for extracting statistical information from complex data. In: Studies in Classification, Data Analysis, and Knowledge Organisation, vol. 15, Springer, Heidelberg (2000)
2. Branigan, H.: Perspectives on multi-party dialogue. Research on Language & Computation 4(2-3), 153–177 (2006)
3. Carriero, N., Gelernter, D., Leichter, J.: Distributed data structures in linda. In: POPL '86. Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 236–242. ACM Press, New York (1986)
4. Dignum, F., Vreeswijk, G.: Towards a testbed for multi-party dialogues. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 212–230. Springer, Heidelberg (2004)
5. Heath, C., Svensson, M.S., Hindmarsh, J., Luff, P., vom Lehn, D.: Configuring awareness. Comput. Supported Coop. Work 11(3), 317–347 (2002)
6. Huget, M.-P., Demazeau, Y.: First steps towards multi-party communication. In: van Eijk, R.M., Huget, M.-P., Dignum, F.P.M. (eds.) AC 2004. LNCS (LNAI), vol. 3396, pp. 65–75. Springer, Heidelberg (2005)
7. Kumar, S., Huber, M.J., McGee, D., Cohen, P.R., Levesque, H.J.: Semantics of agent communication languages for group interaction. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 42–47. AAAI Press/The MIT Press, Cambridge (2000)

An Interaction Protocol for Agent Communication

Gemma Bel-Enguix, M. Adela Grando*, and M. Dolores Jiménez-López

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
{gemma.bel,mariadolores.jimenez}@urv.cat
mariaadela.grando@estudiants.urv.es

Abstract. In this paper, we introduce a formal-language interaction protocol for agent communication that may contribute to the building of better human-computer dialogues through a simulation of human language use. The paper centers on formal dialogue research. We introduce the definition of a formal model of dialogue based on Eco-Grammar Systems (EGS) and inspired in the Multi-Agent Protocol (MAP) language. The result is a simple formal device that could be used for the design of dialogue systems with limited human-like behaviour.

1 Introduction

Computational work on discourse has focused both on extended texts and on dialogues. Work on the former is relevant to document analysis and retrieval applications, whereas research on the latter is important for human-computer interfaces. According to [4], two extreme views on formal dialogue research can be distinguished:

1. the *'engineering' view* that claims that the purpose of formal dialogue research is to enable the building of better human-computer interfaces by incorporating dialogue.
2. and the *'simulation' view* that claims that the ultimate goal of formal dialogue research is a complete computational (implementable) theory of human language use and understanding.

Even though many researchers agree that a complete simulation of human conversation is very difficult (maybe impossible) to be reached, it seems clear that knowledge of human language use can help in the design of efficient human-computer dialogues. Since, human-computer interaction can be conceived as a form of dialogue with a computer, the knowledge coming from areas like conversation analysis may be directly relevant to human-computer interfaces. In fact, human conversation exhibit the natural and 'user-friendly' quality that is a desired feature for human-computer dialogues.

* Suported by research grant "Programa Nacional para la Formación del Profesorado Universitario" from the Ministry of Education, Culture and Sports of Spain.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 62–72, 2007.

In this paper, we introduce the definition of a formal model of dialogue based on Eco-Grammar Systems (EGS) and inspired in the Multi-Agent Protocol (MAP) language. In order to define our model, we have considered some of the features that characterize human language use. In fact, the main feature of the framework introduced here is its interdisciplinarity. To define our formal model we put together knowledge from three different research areas:

- *Artificial intelligence*, since we use multi-agent systems protocols.
- *Formal language theory*, since our model can be seen as a new variant of the so-called grammar systems theory.
- *Linguistics*, since we take into account theories of human language use.

The result is a simple formal device that could be used for the design of effective and user-friendly human-computer dialogues.

Throughout the paper, we assume that the reader is familiar with the basics of formal language theory, for more information see [6].

2 Eco-Grammar Systems and Multi-Agent Protocol

Eco-grammar systems theory is a subfield of grammar systems theory, a consolidated and active branch in the field of formal languages [2]. Eco-grammar systems have been introduced in [3] and provide a syntactical framework for eco-systems, this is, for communities (societies) of evolving agents and their interrelated environment. Briefly, an eco-grammar system is defined as a multi-agent system where different components, apart from interacting among themselves, interact with a special component called 'environment'. So, within an eco-grammar system we can distinguish two types of components *environment* and *agents*. Both are represented at any moment by a string of symbols that identifies current state of the component. These strings change according to sets of evolution rules (Lindenmayer systems). Interaction among agents and environment is carried out through agents' actions performed on the environmental state by the application of some productions (rewriting rules) from the set of action rules of agents.

In multi-agent systems two main approaches to define dialogues can be mentioned: 1) an *agent-centric approach* [11][12] where agents decide their actions dynamically only respecting the restrictions that can derive from their communication model; and 2) a *societal approach* [8] based on the principle that in a society agents typically share a common interest, e.g. solving a particular problem. To become a member of this society, an agent must agree to observe certain rules and conventions expressed as *social norms*. A popular way to express the social norms between groups of agents is by means of an explicit *protocol*, see for instance [10]. A protocol is a statement of the social norms and acts as a guide for the agents to interact without affecting their autonomy. The language provides decision points in which the agents can behave autonomously. These decision points are given by the agent reasoning processes, that we call *decision procedures*, whose output can determine next action to be performed in the protocol.

Eco-grammar systems are biologically inspired in the principle of collaboration between agents in a community. Similarly, the MAP language [9] is based on the societal approach. In MAP the concepts of *scenes* and *roles* are crucial. A *scene* can be thought of as a bounded space in which a group of agents interact on a single task. Formally, a scene S comprises a set of agent *protocols* $P^{(i)}$ which are parameterised on these roles; a set of axioms $K^{(s)}$ known as *common knowledge* that represents the set of facts which are believed to be true for all the agents in the scene; and a set of performatives $M^{(t)}$ which defines all of the allowed performatives for the scene.

Agents participating in a scene adopt a *role* which can be dynamically changed for other role through agent invocations. Associated to the concept of role is the concept of *protocol* which is defined as a sequence of definitions parameterised on the role. Thus, the protocol steps which the agent follows in a scene are directly dependent on the role they assume. The core of the protocols is constructed from agent invocations, control flow operation, null operations, decision procedures and sending and reception of messages. There exists also the possibility of changing operations natural precedence. Through agent invocation, an agent can change its role (same identifier), introduce a new instance of agent (different identifier and role), or implement recursion (same identifier and role). The control flow operations can be either sequences *then* or non-deterministic choices *or*. Interaction between agents is performed by the exchange of *messages*. Messages can be sent or received, specifying their origin or destination. No fix semantics is assigned to these performatives. About *decision procedures* they provide an interface between the communicative and the rational process of the agent. This makes it straightforward to understand the operation of the protocol without knowing procedure implementative details, and makes it possible to verify the protocols using automated means, e.g. model checking. Every agent is defined with a set of axioms corresponding to its private knowledge. We allow decision procedures to access common knowledge and agent private knowledge, but they can only modify private knowledge.

3 Extended Reproductive Eco-Grammar Systems

The MAP protocol language we have introduced in the above section has been designed to model the message interchange in distributed and heterogeneous multi-agent systems, and it has been implemented in JAVA. LCC (Light Coordination Calculus) is a similar approach implemented in Datalog. The main goal of MAP has been to solve problems described by scenes. The language has shown to be expressive enough to express all the scenes it has been used with for problem solving. Strictly, it can model non-human conversation situations for an efficient problem solving by using the parallelism and distribution of agents. The model is very generic. Its main objective is to reach the highest efficiency in the messages interchange. An attempt to introduce linguistic concepts in the protocol can be found in [5] where five generalized patterns for classifying possible message interchanges in human conversation are defined: *response, continuation,*

counter, clarification and *correction*. In [5], the use of these patterns as rewriting rules to generate (synthesize) protocols of the type we present here is suggested.

As we have already said, our goal here is to introduce a framework that can be used to model human-computer conversations, following human-level language understanding. To achieve this goal we have two options to restrict the above protocol:

1. either we keep the syntax of the model while modifying its semantics by fixing the corresponding restrictions,
2. or we define a new model for describing the behaviour of participants in a conversation, with the syntax of the language protocol but with the semantics of human conversations.

We have chosen the second possibility. Therefore, we introduce a new model called *Extended Reproductive Eco-Grammar System* that is based in the Reproductive Eco-grammar Systems introduced in [3]. In this section we introduce the formal definitions of our model.

Definition 1. *An Extended Reproductive Eco-Grammar (EREG) system is an (n+1)-tuple* $\Sigma = (E, \mathcal{A}_1, ..., \mathcal{A}_n)$ *where:*

- $E = (V_E, P_E)$ *is the environment with* V_E *a finite alphabet and* P_E *a finite set of* $0L$ *rewriting rules over* V_E*;*
- $\mathcal{A}_i$ *is a multiset (a set whose elements can occur several times each), called the population of the agents of the i-th type,* $1 \leq i \leq n$*, where every agent* A_i *in* $\mathcal{A}_i$ *has the same form* $A_i = (V_i \cup \{\sqcup\}, P_i, R_i, \varphi_i, \psi_i)$*, where:*
  V_i *is a finite alphabet,* $\sqcup$ *is the reproduction symbol that can occur only on the right-hand side of productions of* P_i*,* P_i *a finite set of* $0L$ *rewriting rules over* V_i*,* R_i *is a finite set of* $0L$ *rewriting rules* $x \rightarrow y$ *over the environment* ($x \in V_E^+$ *and* $y \in V_E^*$) *or over the state of the agent* ($x \in V_i^+$ *and* $y \in V_i^*$)*. Mappings* $\varphi_i : V_E^* \longrightarrow 2^{P_i}$ *and* $\psi_i : V_i^+ \longrightarrow 2^{R_i}$ *are computable functions that select production sets according to the environment state and the agent state, respectively.*

Definition 2. *(System Configuration) A configuration of an EREG system* $\Sigma = (E, \mathcal{A}_1, ..., \mathcal{A}_n)$ *is an (n+1)-tuple:* $\sigma = (\omega_E, W_1, W_2, ..., W_n)$ *where:* $\omega_E \in V_E^*$ *and* W_i *is a multiset of strings* $\omega_{i1}, ..., \omega_{ik_i}$*, where* $\omega_{ij} \in V_i^+$ *for* $1 \leq j \leq k_i$*,* $1 \leq i \leq n$*;* ω_E *is the current evolution state of the environment and the strings* $\omega_{i1}, ..., \omega_{ik_i}$ *correspond to the evolution states of all currently existing agents of the i-th type.*

Definition 3. *(Agent Derivation) Agent state* ω_i *derives to new state* ω_i' *denoted by* $\omega_i \vdash \omega_i'$ *iff* $\omega_i \vdash_1 \beta'$ *as the result of parallel application of all the environment rewriting rules selected by mappings* ψ_i *from agent* A_i *and* $\beta' \vdash_2 \omega_i'$ *by parallel application of all the rules selected by* φ_i*.*

Definition 4. *(Environment Derivation) Environmental state* ω_E *derives to new state* ω_E' *denoted by* $\omega_E \models \omega_E'$ *iff* $\omega_E \models_1 \alpha'$ *as the result of parallel application of all the environment rewriting rules selected by mappings* ψ_i *from all agents and* $\alpha' \models_2 \omega_E'$ *according to* P_E*.*

Definition 5. *(System Derivation) Let* $\Sigma = (E, \mathcal{A}_1, \ldots, \mathcal{A}_n)$ *be an EREG system and let* $\sigma = (\omega_E, W_1, W_2, ..., W_n)$ *and* $\sigma' = (\omega'_E, W'_1, W'_2, ..., W'_n)$ *be two configurations of* Σ*. We say that* σ *is directly changed for (it directly derives)* σ'*, denoted by* $\sigma \Longrightarrow_\Sigma \sigma'$*, iff* ω'_E *arises from* ω_E *by evolution of the environment affected by all agents being active in* σ*, and* W'_i *is the reproduction of the states actually evolving from the states in* W_i*,* $1 \leq i \leq n$*. Note that action rules have priority over evolution rules.*

We have proved that for any scene of message interchange described by the protocol language, it is always possible to simulate in our model the same scene but keeping the human-conversation features enumerated. This is an automatic process. We present here an intuitive theorem, lack of space prevent us from introducing here the whole proof.

Theorem 1. *Given an arbitrary definition of a scene* $S = \langle P^{(n)}, K^{(j)}, M^{(k)} \rangle$ *following the syntax of the protocol language defined above, it can always be defined an EREG system* $\Sigma = (E, \mathcal{A}_1, ..., \mathcal{A}_n)$ *that simulates its behaviour.*

Intuition behind the theorem: The environment $E = (V_E, P_E)$ represents conversational environment shared by all the agents in the scene: protocol definitions $P^{(n)}$, common knowledge $K^{(j)}$ and common set of performatives $M^{(k)}$. Also it represents the physical media through which agents interchange (send or receive) messages and introduce new agents invocations. Therefore, the environment state has the following form:

$\omega_E = ((newagent)^* \langle \rangle_{NA,id} + + (message)^* \langle \rangle_{M,id})^+ + + ShareK + + Protocols + + DF$

- $((newagent)^* \langle \rangle_{NA,id} + + (message)^* \langle \rangle_{M,id})^+$ is used by the agents to introduce new messages and agent invocations. Each agent A_i with identifier id can introduce agent invocations by a rewriting rule that replaces symbol $\langle \rangle_{NA,id}$ by the new definitions followed by $\langle \rangle_{NA,id}$. Messages can also be introduced with the same strategy, but replacing symbol $\langle \rangle_{M,id}$.
- $ShareK$ is the string representation of $K^{(j)}$, which can be accessed but not modified by the agents through the so-called decision procedures.
- $Protocols$ and DF are the string representations of $P^{(n)}$ and $M^{(k)}$, respectively. They correspond to the static part of the environment state, they are defined when starting the simulation and they remain unchangeable.

In each time unit of the evolution process, the parts in the conversation communicate through the environment sending messages and taking from it messages that they receive from the other participants in the conversation. Environment productions P_E are in charge of deleting in each time unit all the information that agents have added to the environment in the previous time unit. This is done with the purpose of simulating what we called *conversational evanescence*. Whenever an agent A_i finds in the environment information that he needs or that has been sent to him, he takes it and copies it in his private mental state ω_i. Strings $Protocols$, $ShareK$ and DF are never deleted from ω_E. The same happens with symbols $\langle \rangle_{NA,id}$ and $\langle \rangle_{M,id}$, needed by agents to introduce new agent instances or to send messages.

Classes $\mathcal{A}_i$, $1 \leq i \leq n$, are defined: one per each possible role in the scene. And every agent $A_i = (V_i \cup \{\sqcup\}, P_i, R_i, \varphi_i, \psi_i)$ that belongs to class $\mathcal{A}_i$ behaves as an agent playing role r_i. The definition of a scene S is completed with a list of agent invocations that start the scene, it means with the list of the individuals starting the conversation. Dynamically during execution time active agents can introduce new instances of individuals, it means they can change their point of view in the conversation (change of role), keep their point of view but restart their participation in the talk (recursion) or call other individual to participate (creation of a new agent instance). The same counts for the model given here: to start the simulation of the scenario S, initial environmental state ω_{E_0} has to contain the list of initial agents. And besides each of the n classes $\mathcal{A}_i$ has to be defined with initial state $W_{i_0} = \{\langle Parent \rangle\}$. If during evolution time agents invocations are introduced in ω_E, an agent from class $\mathcal{A}_i$ with state $\omega_i = \alpha\langle Parent\rangle$ performs a reproductive rule. Its state changes to $\omega_i = \alpha$ and a new agent A_i in $\mathcal{A}_i$ is created with initial state:

$$\omega_i = Identifier + +Memory + +OC + +CState + +PK + +SK\langle Parent\rangle$$

- $Identifier$ is the string representation of the agent identifier id.
- $Memory$ corresponds to the last message received but not processed. So, initially $Memory = \langle\rangle_{IM}$. While the language protocol we present allows each agent to keep a queue of input messages, here we restrict human memory to one message.
- $OC = \langle\downarrow oc\rangle_{OP}$ is the string representation of the operational clause oc assigned to the agent, where symbol $\downarrow$ is used to mark the current operator. During execution time $OC = \langle\alpha \downarrow op_x\beta\rangle_{OP}$ and symbol $\downarrow$ points to the current operator op_x. When $OC = \langle\alpha \downarrow\rangle_{OP}$ the individual is considered inactive, it has no more operations to perform.
- $CState \in \{\langle Close\rangle, \langle Open\rangle, \langle Uncertain\rangle, \langle Inactive\rangle\}$ and $OC = \langle\alpha \downarrow op_x\beta\rangle_{OP}$ are used to know the agent current state. If $CState = Close$ the operator op_x has been applied. If $CState = Open$, op_x has not been applied. If $CState = Uncertain$ and op_x is the sending of a message it indicates that the agent has spoken and does not know if he has been the only speaker. If $CState = Inactive$ and $OC = \langle\alpha \downarrow\rangle_{OP}$ the agent has finished performing his operational clause, what makes him inactive. An agent is initially created with $CState = \langle Close\rangle$ and $OC = \langle\downarrow oc\rangle_{OP}$.
- PK and SK are the strings used for encoding the set of axioms corresponding to the agent private knowledge and a copy of the common knowledge that the agent keeps for himself, respectively.

With respect to mappings φ_i and ψ_i we restrict them to computable functions. We say that $\varphi_i : V_E^* \longrightarrow 2^{P_i}$ simulates the way that an agent listens what is going on in the conversation and also is in charge of generating the corresponding new agent instances in class $\mathcal{A}_i$.

- When finding in ω_E agents invocations with role r_i introduced by agents participating in the scene φ_i applies reproductive rules as described above.

- When detecting that agent A_i has been the only agent speaking φ_i changes $CState$ from $\langle Uncertain\rangle$ to $\langle Close\rangle$ to simulate that what he said could be listened by the agent to whom the message is addressed. But if φ_i finds out that agent A_i has spoken at the same time as other agents, it realizes that an overlapping has taken place. In this case it selects two rules: the first one for changing agent current state to close. And the second rule is undeterministically chosen between the following ones: $\langle \alpha M \Longrightarrow agent(id_h, role_j) \downarrow op_x\beta\rangle_{OP}$ for $\langle \alpha \downarrow M \Longrightarrow agent(id_h, role_j) op_x\beta\rangle_{OP}$ and $\langle \alpha \downarrow (null\, then\, M \Longrightarrow agent(id_h, role_j)) op_x\beta\rangle_{OP}$. In the first case agent decides to repeat what he said before. While in the second case agent first listens, allowing other agents to speak, and then speaks. In this way a repairing mechanism is simulated.
- When discovering that agent A_i has received one message from an agent who has been the only speaker, φ_i adds the message to the agent memory. But, first it is checked that the received message is valid, i.e. the performative is from the dialogical framework DF in ω_E. In case φ_i finds out that more than one agent has spoken, the message is not copied to agent memory, to simulate that if more than one agent speaks the agent can not distinguish any message in the conversation.

$\psi_i : V_i^+ \longrightarrow 2^{R_i}$ is used to simulate speaker mental changes that decides how the agent participates in the scene speaking and changing the conversational environment.

- Unconditionally, in every time unit it simulates the evanescence of human memory deleting any message that is in the agent's memory, keeping just the symbol $\langle\rangle_{IM}$.
- ψ_i is in charge of simulating the application of operators *then*, *or*, *decision procedures*, the *sending of a message* and the *processing of a received message*. According to the substrings $OC = \langle \alpha \downarrow op_x\beta\rangle_{OP}$ and $CState \in \{\langle Close\rangle, \langle Open\rangle, \langle Uncertain\rangle, \langle Inactive\rangle\}$ from ω_i mapping ψ_i determines the current operator and tries to apply it:
  - If $CState = \langle Uncertain\rangle$ then op_x is the sending of a message. In this case mapping ψ_i realizes that the agent has spoken but he does not know if other agent has talked at the same time, so he does not change the state or the position of pointer $\downarrow$.
  - If $CState = \langle Close\rangle$, ψ_i interprets that marker $\downarrow$ is pointing an operator already executed and that it should try to apply next operator in $\langle \alpha \downarrow op_x\beta\rangle_{OP}$. If next operator op_y can be applied, mapping ψ_i selects from R_i a set of rules according to the type of op_y. For example, if $op_y = (M \Longrightarrow \mathbf{agent}(id_h,\ r_j))$, ψ_i tries to simulate the sending of a message. The new message is introduced in the conversational context replacing symbol $\langle\rangle_{M,id}$ by string $\langle M, [id, i], [id_h, j]\rangle_{M,id}\langle\rangle_{M,id}$, where $[id, i]$ represents the sender and $[id_h, j]$ denotes the receiver. And the agent state is changed from $\langle Close\rangle$ to $\langle Uncertain\rangle$.
  - If there are no more operators to apply, $OC = \langle \alpha \downarrow\rangle_{OP}$, then the state of the agent has to be set as inactive. Mapping ψ_i selects from R_i the set of

rules: $\{\langle Close\rangle \rightarrow \langle Inactive\rangle\}$. In the next derivation step, the system detects that the agent is inactive and deletes all the symbols from its state ω_i, except the symbol $\langle Parent\rangle$. If the agent was in charge of creating new instances of agents of the same class ($\omega_i = \beta\langle Parent\rangle$) it is not considered any more an active agent but keeps the duty of procreator until it performs the first introduction of new agent instances. As it was described before, the creation of a new agent of the class leaves the agent state in an empty one (ω_i=λ). An agent with state ω_i=λ is inactive and the system eliminates it in the next derivation step.

- If $CState = \langle Open\rangle$ and $OC = \langle\alpha \downarrow op_x\beta\rangle_{OP}$, ψ_i interprets that marker $\downarrow$ is pointing an operator op_x that has not yet been executed and tries to apply it.
- If $OC = \langle\alpha(\downarrow op_x \; or \; op_y)\beta\rangle_{OP}$ or $OC = \langle\alpha(op_y \; or \downarrow op_x)\beta\rangle_{OP}$, ψ_i makes some kind of backtracking. It changes the current state to close and rewrites OC to $\langle\alpha(op_x \; or \downarrow op_y)\beta\rangle_{OP}$ or $\langle\alpha(\downarrow op_y \; or \; op_x)\beta\rangle_{OP}$, respectively. In this way a situation of deadlock for agent A_i is avoided, allowing to try with the other term of the *or* operator in next derivation step.

Notice that in order to define this new model we have considered some of the features that characterize human-human conversations (cfr. [1] and [7]):

- *Copresence.* Participants share the same physical environment. In our model this is the scene, a set of performatives or dialogical framework, and axioms or common knowledge
- *Audibility.* Participants can hear each other. We simulate speaking by ψ and we differentiate between the physical listening of the message by φ and the act of processing, interpreting a message that has been listened, by ψ.
- *Instantaneity.* Participants perceive each other's actions at no perceptible delay. In our case delay is one time unit. An agent speaks in time unit t_0 and the receiver listens the message at time $t_0 + 1$.
- *Evanescence.* Medium is evanescent, it fades quickly. In our model P_E deletes what agents introduce to the environmental previous time unit. Or if an agent does not want to pay attention to what other agent says, he hears the message in one time unit and delete it (forget it) next time unit.
- *Simultaneity.* Participants can produce and receive at once and simultaneously. Because φ and ψ are applied in parallel, simultaneity is possible.
- *Extemporality.* Participants formulate and execute their actions extemporaneously, in real time. For us time is given by a common clock.
- Overwhelmingly, *one party talks at a time.* We allow n$\geq$ 0 agents to speak simultaneously.
- *Occurrences of more than one speaker at a time* are common, but brief. In our model we allow n$\geq$ 2 agents to speak, but in this case no agent listens and a repair mechanism takes place.

- *Turn order and turn size are not fixed*, but varies; *Length of conversation and what parties say are not specified in advance.* In our model all these features depend on decision procedures and undeterministic choices.
- Relative *distribution of turns is not specified in advance.* It depends on interaction, distribution and agent decisions.
- *Number of parties can vary.* In our model it is possible because of being a variant of reproductive eco-grammar system.

4 Example: A Selling Process

In this section, we exemplify the use of the MAP language with the scene corresponding to a simplification of a selling process in a bookshop. The example comprises four protocol definitions: 1) *Clerk*, 2) *Cmer* for Customer, 3) *Seller* and 4) *Buyer*, but here we only show the *Clerk* and *Customer* protocol definitions. A customer asks for a book '$book_1$' to a clerk. If the clerk answers him that the book is not in existence he can decide either to ask for another book '$book_2$' or to exit the scene. In case '$book_1$' is in existence he can buy it, adopting the role of a buyer. Symmetrically, the clerk is in charge both of checking the existence of the book the customer is looking for and of informing him of the result of that process. In case the book is in existence, he leaves a seller in charge of the selling process and he goes on attending other customers.

$$\begin{array}{l|l}
\mathbf{agent}(\mathbf{A}, \mathbf{Cmer}, \mathbf{K_{Cu}}, \mathbf{book}_1) = & \mathbf{agent}(\mathbf{B}, \mathbf{Clerk}, \mathbf{K_{Cl}}) = \\
ask(book_1) \Longrightarrow agent(B, Clerk)\ then & ask(book) \Longleftarrow agent(A, Cmer)\ then \\
 & answer = check(book, K_{Cl})\ then \\
\quad deny() \Longleftarrow agent(B, Clerk)\ then & \\
\quad\quad book_2 = newsearch(K_{Cu})\ then & \quad deny() \Longrightarrow agent(A, Cmer) \\
\quad\quad agent(A, Cmer, \varnothing, book_2 & \quad then\ agent(B, Clerk, \varnothing) \quad or \\
\quad or\ \epsilon & \\
or \quad accept() \Longleftarrow agent(B, Clerk) & \quad accept() \Longrightarrow agent(A, Cmer) \\
\quad then\ agent(A, Buyer, \varnothing, book_1) \quad . & \quad then\ sell(book) \Longrightarrow agent(C, Seller) \quad . \\
 & \quad then\ agent(B, Clerk, \varnothing)
\end{array}$$

This protocol definition can be used in a web application like Amazon, where one clerk agent can receive thousands of customers' requests simultaneously. According to MAP semantics, in this kind of multiagent environment this situation is possible and it is implemented by the clerk agent keeping a list of input messages. But in our definition, because we model human-like conversations, turn-taking is warranted. So, if more than one customer requests for books simultaneously, mapping φ_{Clerk} detects it in ω_E and does not actualise clerk state ω_{Clerk}. In next derivation step, a repairing mechanism is applied. All customers detect that because if they spoke simultaneously clerk did not hear them. So, each of them chooses, in an autonomous and undeterministic way by mapping $\psi_{Customer}$, to reintroduce the same request or to be quiet and give other customer the chance to speak.

5 Final Remarks

In this paper, we have introduced Extended Reproductive Eco-Grammar Systems as a formal model of dialogue that is based on eco-grammar systems and inspired in the MAP language. EREG offers a model of dialogue with a high degree of flexibility where strings can be modified during running time, allowing to dynamically alter agent behaviour according to the context changes. Taking into account that human-computer interfaces require models of dialogue structure that capture the variability and unpredictability within dialogue, we think that EREG may offer a useful formalism for the field of computer dialogue systems.

The model introduced here is just an initial approximation to the possibility of formalizing dialogue by means of a formal language model. For instance the formal model we introduce here inherits from MAP language the restriction that the knowledge shared by all the agents in a dialogue can be freely accessed but not modified. Since Extended Reproductive Eco-Grammar Systems provide an evolving share environment we can consider to study the dynamics of the model after removing the mentioned restriction. Many important aspects remain to be approached, but we think that a more intensive work on this topic may provide a powerful and interesting model of dialogue that conjugates the formal nature, flexibility and naturalness of eco-grammar systems with the computational nature of the MAP language. Such a model of dialogue will have the explicitness, formality and efficiency that are required for the implementation of human-computer dialogues.

References

1. Clark, H.H.: Using Language. Cambridge University Press, Cambridge (1996)
2. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, Gh.: Grammar Systems: A Grammatical Approach to Distribution and Cooperation, Gordon and Breach (1994)
3. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, Gh.: Eco-Grammar Systems: A Grammatical Framework for Life-Like Interactions. Artificial Life 3(1), 1–28 (1996)
4. Larsson, S.: Dialogue Systems: Simulations or Interfaces? In: Gardent, C., Gaiffe, B. (eds.) DIALOR'05. Proceedings of the Ninth Workshop on the Semantics and Pragmatics of dialogue, pp. 45–52 (2005)
5. McGinnis, J., Robertson, D., Walton, Ch.: Protocol Synthesis with Dialogue Structure Theory. In: Parsons, S., Maudet, N., Moraitis, P., Rahwan, I. (eds.) ArgMAS 2005. LNCS (LNAI), vol. 4049, Springer, Heidelberg (2006)
6. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages. Springer, Heidelberg (1997)
7. Sacks, H., Schegloff, E.A., Jefferson, G.: A Simplest Systematics for the Organization of Turn-Taking for Conversation. Language 50(4), 696–735 (1974)
8. Shoham, Y., Tennenholtz, M.: On Social Laws for Artificial Agent Societies: Off-Line Design. Artificial Intelligence 73, 231–252 (1995)
9. Walton, Ch.: Multi-Agent Dialogue Protocols. In: Proceedings of the Eighth International Symposium in Artificial Intelligence and Mathematics (2004)

10. Esteva, M., Rodriguez, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the Formal Specification of Electronic Institutions. In: Dignum, F.P.M., Cortés, U. (eds.) Agent-Mediated Electronic Commerce III. LNCS (LNAI), vol. 2003, pp. 126–147. Springer, Heidelberg (2001)
11. Greaves, M., Holmback, H., Bradshaw, J.: What is a Conversation Policy? Issues in Agent Communication, pp. 118–131. Springer, Heidelberg (2000)
12. Rao, A.S., Georgeff, M.: Decision procedures for BDI logics. Journal of Logic and Computation 8(3), 293–344 (1998)

Collaborative Attack Detection in High-Speed Networks

Martin Rehák[1], Michal Pěchouček[2], Pavel Čeleda[3], Vojtěch Krmíček[3], Pavel Minařík[3], and David Medvigy[2]

[1] Center for Applied Cybernetics, Faculty of Electrical Engineering
[2] Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague Technická 2, 166 27 Prague, Czech Republic
{mrehak,pechouc}@labe.felk.cvut.cz
[3] Institute of Computer Science, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{celeda,vojtec}@ics.muni.cz

Abstract. We present a multi-agent system designed to detect malicious traffic in high-speed networks. In order to match the performance requirements related to the traffic volume, the network traffic data is acquired by hardware accelerated probes in NetFlow format and preprocessed before processing by the detection agent. The proposed detection algorithm is based on extension of trust modeling techniques with representation of uncertain identities, context representation and implicit assumption that significant traffic anomalies are a result of potentially malicious action. In order to model the traffic, each of the cooperating agents uses an existing anomaly detection method, that are then correlated using a reputation mechanism. The output of the detection layer is presented to operator by a dedicated analyst interface agent, which retrieves additional information to facilitate incident analysis. Our performance results illustrate the potential of the combination of high-speed hardware with cooperative detection algorithms and advanced analyst interface.

1 Introduction

The purpose of the presented work is to deliver an autonomous system able to detect malicious traffic on high-speed networks and to alert the operators efficiently. While the system reasoning is based on intelligent agents and multi-agent methods, the network traffic data acquisition and preprocessing in both dedicated adaptive hardware and specialized software is essential for project success, as the traditional agent techniques are not well suited for efficient low-level traffic processing.

In the work presented in this paper, we aggregate the network data to capture the information about network flows, unidirectional components of TCP connections (or UDP/ICMP equivalent) identified by shared source and destination addresses and ports, together with the protocol, and delimited by the time frame used for data acquisition (see Section 2.1). This information provides no hint about the content of the transmitted data, but by detecting the anomalies in the list of flows acquired over the monitoring period, we can detect the anomalies and possible attacks, albeit with limited effectiveness.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 73–82, 2007.

In order to detect an attack from the flow information on the backbone level, especially without any feedback from the affected hosts, we have to analyze the patterns in the traffic data, compare them with normal behavior and conclude whether the irregularity corresponds to a known attack profile or not. This approach to Network Intrusion Detection, typically based on the flow information captured by network flow monitor is currently an important field of research into *anomaly based intrusion detection*. Numerous existing systems, based on traffic volume analysis modeled by Principal Component Analysis methods [1], models of entropy of IP header fields for relevant subsets of traffic [2,3], or just count of the flows corresponding to the selected criteria [4] offer each a particular valid perspective on the network traffic. In our approach, we have decided not to develop a novel detection method, by rather to integrate each of the methods with an extended trust models of a specialized agent. This combination allows us to correlate the results of the used methods and to combine them to improve their effectiveness. Most anomaly detection methods today are not fit for commercial deployment due to the high ratio of false positives (i.e. legitimate traffic classified as malicious) or false negatives (malicious traffic classified as legitimate). While their current level of performance is a valid scientific achievement, the costs associated with supervision of such systems are prohibitive for most organizations. Therefore, our main research goal is to combine the efficient low-level methods for traffic observation, with multi-agent detection process to detect the attacks with comparatively lower error rate (see Table 1), and to provide the operator with efficient incident analysis layer presented in Section 2.3. Analysis layer supports operator's decisions about detected anomalies by providing additional information from related data sources. It is also responsible for visualization of the anomalies and the detection layer status.

2 Architecture

This section details the design of the system architecture. We present an overview of key techniques and technologies for each layer, and we also present the motivations behind the selection of individual techniques or methods.

The architecture consists of several layers with varying requirements on on-line processing characteristics, level of reasoning and responsiveness. While the low-level layers need to be optimized to handle network traffic at wire speed during the traffic acquisition and preprocessing, the higher layers will use the preprocessed data to infer the conclusions regarding the degree of anomaly and consecutively also the maliciousness of the particular flow or a group of flows. Therefore, while the computation in the higher layers must be still reasonably efficient, the preprocessing by the lower layers allows us to deploy more sophisticated algorithms.

System can be split into these layers, as shown in Figure 1:

2.1 Traffic Acquisition and Preprocessing Layer

The traffic acquisition and preprocessing layer is responsible for acquiring network traffic, preprocessing data and providing traffic characteristics to upper system layers. We use the flow characteristics, based on information from packet's headers.

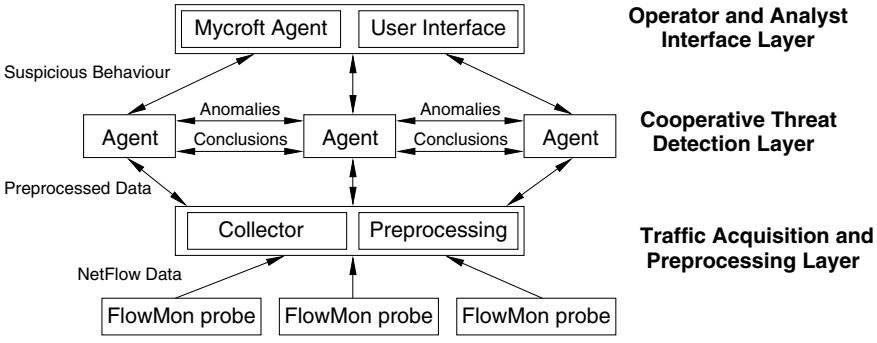


Fig. 1. System overview, with network probes, acquisition and preprocessing layer at the bottom, agent platform for anomalies detection in the middle and visualization on the top

In general, flows are a set of packets which share a common property. The simplest type of flow is a 5-tuple, with all its packets having the same source and destination IP addresses, port numbers and protocol. Flows are unidirectional and all their packets travel in the same direction. For the flow monitoring we use NetFlow protocol developed by Cisco Systems.

The amount of traffic in nowadays high-speed networks increases continuously and traffic characteristics change heavily in time (network throughput fluctuation due to time of day, server backups, DoS attacks, scanning attacks, etc.). Performance of network probes must be independent of such states and behave reliably in all possible cases. The quality of provided data significantly effects the upper layers and chances to detect traffic anomalies.

Therefore we use hardware accelerated NetFlow probes FlowMon [5] a passive network monitoring device based on the COMBO hardware [6], which provides high performance and accuracy. The FlowMon probe is preferred due to implemented features which contains packet/flow sampling, several sampling types, active/inactive timeouts, flow filtering, data anonymization, NetFlow protocol version 5 and 9 support. The FlowMon probe handles 1 Gb/s traffic at line rate in both directions and exports acquired NetFlow data to different collectors. Detailed evaluation of these crucial capabilities is described in Section 3.

The collector stores incoming packets with NetFlow data from FlowMon probes into database. The collector provides interface to graphical representation of network traffic, flow filtration, aggregation and statistics evaluation, using source and destination IP addresses, ports and protocol.

To process acquired IP flows by upper system layers the preprocessing must be performed on several levels and in different manners. Packets can be sampled (random, deterministic or adaptive sampling) on input and sampling information is added to NetFlow data. On the collector side the same flows are aggregated to reduce the amount of data without information loss and several statistic characteristics (average traffic values, entropy of flows) are computed.

After deploying probes in monitored network, the probes can be reprogrammed to acquire new traffic characteristics. The system is fully reconfigurable and the probes can

adapt their features and behavior. As we can see in Section 3, presented solution can acquire unsampled flow data from even very fast links. The proposed traffic acquisition and preprocessing layer is able to provide real-time traffic characteristics to detect anomalies by upper system layers.

2.2 Cooperative Threat Detection Layer

Cooperative threat detection is based on the principles of trust modeling [7] that are an integral part of agent research. However, there are three important features [8] that must be added to trust modeling to cover our domain-specific requirements:

• **Uncertain Identity Modeling:** Baseline trust models evaluate the behavior of individual agents, whose identity is guaranteed (to an extent) by the multi-agent platform or similar computational environment. In the network domain, we have to evaluate the trustfulness of network flows, and while they can be distinguished as unique identities, this distinction is unpractical from the intrusion detection perspective. We represent the connections in a metric space, and use the associated distance function to assess the similarity of the flow representations in this space. All detection agents use the same NetFlow quintuple to construct the identity representations, but may obtain different results regarding the similarity due to the use of different distance functions. For example, one agent can emphasize the similarity of srcIP address (and likely host), while the others may concentrate on ports that are more application specific. This variability makes the agent perspective on the system multi-faceted, and the attacks are less likely to avoid multiple agents.

• **Context Modeling:** Merely representing the flow identities in the metric space and evaluating their trustfulness gives unsatisfactory results, as it ignores the most important information from the NetFlow data – the information about the related flows in the current traffic sample. This information constitutes the context of the trusting decision [9], and together with the identity defines the Identity-Context metric space, where the detection agents assess the trustfulness of flow representations. Each of the agents uses its own particular context space, typically based on the existing anomaly detection methods. For instance, we can complement the information about the flow by the number of flows from the same srcIP and same dstPrt, or with an entropy of dstIP addresses computed over all flows with the same srcIP. The use of this information is twofold; each agent uses it to place the flow representations in the Identity-Context space of its trust model, and in the same time to provide the information about the degree of flow anomaly to other trusting agents.

• **Implicit Feedback Mechanism:** The principal input of classic trust models is a result of past cooperations with the partner: quality of service, degree of success, on-time delivery and other domain specific parameters. In our case, the system is deployed on backbone network and it is very difficult to obtain the feedback that can be associated with the current traffic on the network; cooperative IDS (like `dshield.org`) typically provide unsynchronized feedback, and not all the threats are Internet-wide. Obtaining the feedback from the connected operators or organizations is even more difficult: while the IETF has several working groups focusing on incident response interoperability, the bulk of the work is not suitable for real-time data processing, and concentrates on human-

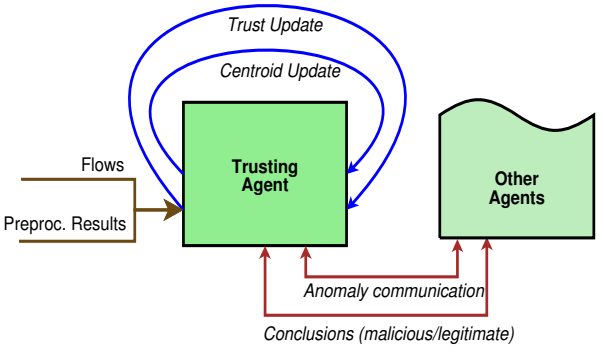


Fig. 2. Overview of detection (trusting) agent operations

to-human interaction. Therefore, we use the information regarding the flow anomaly *as assessed by the other agents* to replace the direct feedback, therefore connecting the anomaly detection between diverse agents.

While processing the information about the network flows, each trusting agent receives an identical copy of network flows list and associated pre-extracted statistics. Then, it uses its specific preprocessing to determine the anomaly of each flow (in most cases working only with already extracted statistics) and to communicate the list of anomalies to other agents. In its turn, the agent also receives the anomalies from the others, and starts the flow processing by its internal trust model. As we have implicitly suggested above, the trustfulness is not associated to individual flows, but rather to selected objects in the Identity-Context space. Individual flow is therefore represented by its identity (i.e. NetFlow quintuple) and the associated context is retrieved to determine its position in the Context subspace. Then, we retrieve the positions of nearby centroids from the current trust models and update their trustworthiness with an aggregated degree of flow anomaly as determined by the other agents. When there is no appropriate cluster in the vicinity of the observed flow, a new cluster is created. The details of the approach are presented in [8].

The decision whether a given flow is trusted or untrusted depends on the typical degree of anomaly in the observed network traffic. This parameter varies widely with network type – the number of anomalies is typically low on corporate or government WANs, but is significantly higher on public Internet backbone, or in the university networks. To avoid the problems with manual tuning of the system, we use a fuzzy-inference process integrated with the trust model to decide whether the given flow is malicious, by computing its inference with Low and High trust values. These values are determined similarly to the trustfulness of individual flow representations, but are represented as two fuzzy intervals [10].

2.3 Operator and Analyst Interface Layer

The Cooperative Threat Detection Layer is coordinated by a super-agent Mycroft. Mycroft is again a multi-agent system. This system is constructed for context based inference over information synthesized from various data sources. The name of this system refers

to the so called Mycroft problem well known from Doyle's Mycroft Holmes - a Sherlock Holmes' brother. Every detected suspicious behavior on the network is reported to the agent Mycroft by the detection layer. Mycroft opens the new case subsequently and retrieves relevant information from data sources to which it is connected. Then the network operator can explore and evaluate the reported case together with contextual information retrieved from connected data sources. One of the main Mycroft's abilities is the adaptability which consists in:

- integration of data sources relevant to observed network,
- adaptation to detection layer status,
- adaptation to current traffic situation on the network,
- personalization for given network operator.

All these adaptations are possible thanks to the following construction: Individuals are classified into categories to express their properties. The individual can be any object relevant to the suspicion detection or any elementary relation between such objects relevant to the suspicion detection. Individuals are classified into categories with a measure from the interval $< -1, 1 >$. Value -1 stands for "certainly not in given category", value 1 stands for "certainly is in given category" and value 0 stands for "cannot decide". This classification is called the elementary fact and is realized always in some context. The context is nothing else than a set of elementary facts. All the information retrieved from data sources is disassembled into elementary facts. This approach allows data utilization in a way that fits actual situation. Presented construction is a straight extension of the so called diamond of focus presented in [11] for the first time. Formal definition of systems referred to as Knowledge and Information Robots is provided in [12]. Multi-agent system Mycroft is one of the representatives of Knowledge and Information Robots class.

Different data sources can be used to support network operator's evaluations and decisions. Interoperability with data sources means that The multi-agent system Mycroft can be taught what kind of information given data source contains, how to combine this information with all other information that system already knows or is capable to acquire and how to access this information. Communication with data source is realized by the set of technical adapters. Learning methods are used throughout the Mycroft agent to provide the adaptation functionality. Teaching is realized using quite formalized natural language instead of manual customization and programming.

Adaptability on the detection layer status means that the multi-agent system Mycroft monitors current status of each agent present in the detection layer and is able to communicate with him. Main purpose of this communication is agent reconfiguration according to operator's demands, current situation in the monitored network or some other specified rules. These rules are taught in the same way as data sources using quite formalized natural language and can be modified.

Adaptability on the current network traffic means that the multi-agent system Mycroft can choose suitable data sources and provide additional knowledge to support operator's decisions. Using additional knowledge provided by these data sources Mycroft can perform basic evaluations of the traffic situation and present this evaluations to the network operator concurrently with the detection report. In some basic cases, using prelearned transmutation patterns, system can even evaluate this suspicion as false positive

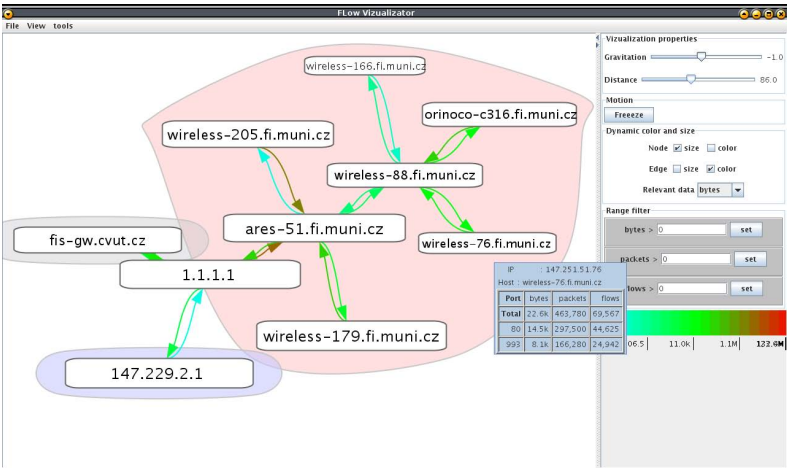


Fig. 3. Mycroft's flow data visualization example. Oval nodes in graph represent network devices identified by IP addresses. Arrows show a direction of the flows.

and drop it. Again this is possible by the means of the knowledge provided by suitable data sources.

Personalization for given operator means that the system is able to follow operator's procedures, habits and preferences. System learns during routine work. When operator is not satisfied he can ask the system for different presentation of given information.

3 System Evaluation and Performance

The performance is becoming a key concern of NIDS *(Network Intrusion Detection Systems)* in high-speed networks. The results of traffic acquisition and processing vary depending on the amount of acquired data. Numerous existing NIDS are based on commodity hardware with open-source software and very limited hardware acceleration.

The Figure 4 shows flow statistics for various IP packet's sizes transmitted on Ethernet at a rate of a gigabit per second. The tests were performed with the Spirent AX/4000 broadband test system [13]. The test system generated 5 flows with 500000 packets per flow for each packet size. The results of FlowMon probe correspond to the maximal theoretical throughput on gigabit Ethernet network. On the other hand the results of software NetFlow probe (nProbe [14]) are misrepresented for small IP packets. The nProbe was used on Linux OS with Intel PCI-X network interface card and default kernel configuration.

Existing flow monitoring systems are mostly based on exporting flow data from routers. The routers are dedicated for routing the data in networks and enabling the flow export has often negative impacts on overall router performance especially during attacks. The exported flows are sampled or limited to maximal number of exported

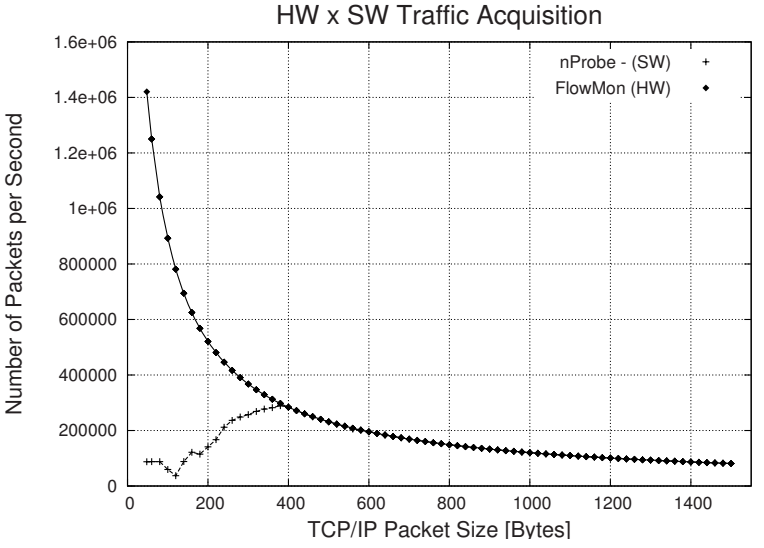


Fig. 4. Flow statistics acquired by SW and HW accelerated probe

flows per second e.g. 5000 flows/s. In comparison with FlowMon probe, user defined extensions can't be added to the routers and the adaptability is very limited.

In our work we are focusing on the impact of packet sampling on anomaly detection. The articles [15,16] study whether existing sampling techniques distort traffic features that are critical for effective anomaly detection. They show that packet sampling methods (random packet sampling, random flow sampling, smart sampling, and sample-and hold sampling) introduce fundamental bias that degrades the performance of the anomaly detection algorithms. To avoid such a misbehavior the FlowMon probe provides non-sampled data, without packet loss at a line rate.

The Table 1 shows the performance of multi-agent system. The observed network traffic is processed by several layers to handle high amount of data in nowadays networks. Each layer has specific physical and performance limits e.g. number of incidents which can be handled by human operator. To overcome such limitations, the system is fully scalable and all layers can be distributed. The multi-agent system adapts the detection behavior to reduce the number of false positives and negatives so the final number of incidents fits the limits of human operator.

4 Conclusions and Future Work

Our work presents a design of multi-agent system for network intrusion detection that is optimized for deployment on backbone networks. Our system addresses two main limitations of existing intrusion detection systems – efficiency and effectiveness. Deployment on high-speed links implies the need to process the important quantity of

Table 1. Multi-agent system performance overview. The system process backbone link traffic with average load of 800 Mb/s.

Layer	*Processed Data*	
	Input	*Output*
Operator	Security Incidents - incidents at a certain priority levels	Incident Handling - resolving up to 10 high priority incidents per hour
Operator and Analyst Interface Layer	Network Anomalies - detected threads with additional network information	Detected Incidents - priority-based incidents up to 100 incidents/minute
Cooperative Threat Detection Layer	Network Traffic Statistics - aggregated flow statistics up to 100000 flows/minute	Detected Threats - network traffic anomalies up to 10000 threats/minute
Traffic Acquisition and Preprocessing Layer	Network Traffic - packets 125000 packets/s	Flow Statistics - flows 3800 flows/s

data in near real-time, in order to prevent the spread of novel threats. Therefore, the individual agents do not acquire the data from the network directly, but receive the data already preprocessed, with the level of detail that is appropriate for anomaly-based intrusion detection. Each detection agent in the system is based on existing anomaly detection technique, which defines its perception of network flow identities in its trust model. Its private view of the data is complemented by the opinions of other agents regarding the anomaly of flows in the current traffic, therefore collaboratively improving the effectiveness of anomaly detection process. When the agents reach a conclusion regarding the untrustfulness of a particular subset of flows, they submit this observation to user-interface agent that automatically retrieves context information (DNS records, history, etc.) to allow rapid analysis by human supervisors.

At the time of this writing, the first preliminary version of the complete system is being integrated and the whole system is still under active development. Therefore, we don't present any definitive experimental results regarding its effectiveness of the complete system, but only the performance evaluations of critical components at the current integration stage. The data used for system testing are acquired on Masaryk University network, connected to the Czech national educational network (CESNET). In our future work, we will provide detailed experimental evaluation of system deployment, and also analyze its performance in countering a wide selection of currently observed malicious traffic.

Acknowledgment. This material is based upon work supported by the European Research Office of the US Army under Contract No. N62558-07-C-0001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Research Office of the US Army. Also supported by Czech Ministry of Education grants 1M0567 and 6840770038.

References

1. Lakhina, A., Crovella, M., Diot, C.: Characterization of Network-Wide Anomalies in Traffic Flows. In: ACM SIGCOMM conference on Internet measurement IMC '04, pp. 201–206. ACM Press, New York (2004)
2. Xu, K., Zhang, Z.L., Bhattacharrya, S.: Reducing Unwanted Traffic in a Backbone Network. In: USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI), Boston, MA (2005)
3. Lakhina, A., Crovella, M., Diot, C.: Mining Anomalies using Traffic Feature Distributions. In: ACM SIGCOMM, Philadelphia, PA, August 2005, pp. 217–228. ACM Press, New York (2005)
4. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J., Dokas, P.: MINDS - Minnesota Intrusion Detection System. In: Next Generation Data Mining, MIT Press, Cambridge (2004)
5. Čeleda, P., Kováčik, M., Koníř, T., Krmíček, V., Špringl, P., Žádník, M.: FlowMon Probe. Technical Report 31/2006, CESNET, z. s. p. o. (2006), `http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe/`
6. CESNET, z. s. p. o.: Family of COMBO Cards (2007), `http://www.liberouter.org/hardware.php`
7. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24, 33–60 (2005)
8. Rehak, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Klusch, M., Hindriks, K., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS(LNAI), vol. 4676, pp. 298–312. Springer, Heidelberg (2007)
9. Rehak, M., Gregor, M., Pechoucek, M., Bradshaw, J.M.: Representing context for multiagent trust modeling. In: IAT'06. IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings), Los Alamitos, CA, USA, pp. 737–746. IEEE Computer Society, Los Alamitos (2006)
10. Rehák, M., Foltýn, L., Pěchouček, M., Benda, P.: Trust Model for Open Ubiquitous Agent Systems. In: Intelligent Agent Technology. 2005 IEEE/WIC/ACM International Conference, vol. PR2416, IEEE, Los Alamitos (2005)
11. Staníček, Z.: Universal Modeling and IS Construction. PhD thesis, Masaryk University, Brno (2003)
12. Procházka, F.: Universal Information Robots a way to the effective utilisation of cyberspace. PhD thesis, Masaryk University, Brno (2006)
13. Spirent, C.: Spirent AX/4000 Broadband Test System (2007), `http://www.spirentcom.com/`
14. Deri, L.: nProbe - An Extensible NetFlow v5/v9/IPFIX GPL Probe for IPv4/v6 (2007), `http://www.ntop.org/nProbe.html`
15. Mai, J., Chuah, C.N., Sridharan, A., Ye, T., Zang, H.: Is sampled data sufficient for anomaly detection? In: IMC '06. Proceedings of the 6th ACM SIGCOMM on Internet measurement, pp. 165–176. ACM Press, New York (2006)
16. Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A.: Impact of packet sampling on anomaly detection metrics. In: IMC '06. Proceedings of the 6th ACM SIGCOMM on Internet measurement, pp. 159–164. ACM Press, New York (2006)

Commitment Monitoring in a Multiagent System

Paola Spoletini[1] and Mario Verdicchio[2]

[1] Politecnico di Milano
spoletin@elet.polimi.it
[2] Università di Bergamo
mario.verdicchio@unibg.it

Abstract. Agent Communication Languages (ACLs) play a fundamental role in open multiagent systems where message exchange is the main if not the only way for agents to coordinate themselves. New proposals about ACL semantics based on social commitments aim at countering the shortcomings of the mainstream mental-state-based ones. The commitment solution does not come for free and calls for an adequate monitoring system that checks whether commitments are fulfilled or not.

1 Introduction

Agent Communication Languages (ACLs) play a very significant role in the context of multiagent systems, especially open ones, in which every agent's internal architecture has been independently developed and so the only common ground it can share with other agents in the system is a standard communication framework. The most important ACL proposal so far, FIPA ACL [9], has gone through some changes, but since its first version it has provided language specifications in terms of agents' mental states. It has already been discussed elsewhere [6,15] how several issues rise from this approach. Let us just remind here that standards relying on a theory involving mental states compel agent programmers to create software with a specific architecture, implementing such theory and thus enabling agents to communicate. In open multiagent systems, populated by heterogeneous, independently developed agents, communication standards should be specified so that agents' internal architecture need not be taken into account. The issues that probably hindered the universal success of mental state-based proposals, also triggered an alternative approach. Some researchers have proposed a semantic framework for an ACL based upon the concept of social commitment [13,5], in which a communicative act is viewed as an action performed by an agent to create or modify the commitments by which it is bound to the others. Commitments, as opposed to mental states, are objective and public and thus have the great advantage of not needing to be reconstructed and attributed to other agents by means of inference processes, and they can be stored in public records for further reference.

2 The Formal Model

Commitments describe certain states of affairs at different instants, thus we need some formal definitions dealing with time, events, and actions. We provide here

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 83–92, 2007.

a restricted version of a temporal logic that has already been illustrated in [15]. In this previous work the temporal model was relying on a frame branching in the future, because the aim was to deal with the truth conditions of future directed sentences without the need for departing from a classic two-valued logic. Instead, we take a more operational approach, and only evaluate the truth value of past and present directed portions of sentences, while keeping the future directed parts suspended for further consideration. Moreover, since in this work we propose a technique for monitoring, that is, checking temporal and atemporal properties of a multiagent system at runtime, we only need to focus on a single run of the system. With such working hypotheses, a branching model of time is superfluous, and we can simply make do with a linear one. Thus, our starting point is LTL$^{\pm}$, a temporal language close to LTL, a linear temporal logic including only future-directed temporal operators [8], to which we add past-directed operators, which allow us to express some properties of computational systems in a far more succinct way [12]. In LTL$^{\pm}$, time is assumed to be discrete, with a starting point but no ending point.

We rely on the classical LTL$^{\pm}$ semantic definitions, and take a model M as comprised of a set of states S, an infinite sequence x of states $(x(0), x(1), ... \in S)$, and a function v that labels each state with the atomic formulae true at that state [8]. Given a model M, an atomic formula ϕ holds at a state $x(t)$ if and only if $\phi \in v(x(t))$. The semantics of the following compound formulae is defined in the usual way: $\neg\phi$ (not ϕ), $\phi \wedge \psi$ (ϕ and ψ), $\mathsf{X}^{+}\phi$ (at the next state ϕ), $\mathsf{X}^{-}\phi$ (at the previous state ϕ), $\phi\mathsf{U}^{+}\psi$ (ϕ until ψ), and $\phi\mathsf{U}^{-}\psi$ (ϕ since ψ). We use the usual derived logical connectives and temporal operators: $\phi \vee \psi$ (ϕ or ψ), $\phi \rightarrow \psi$ (if ϕ then ψ), $\phi \leftrightarrow \psi$ (ϕ iff ψ), $\mathsf{F}^{+}\phi$ (sometimes in the future ϕ), $\mathsf{F}^{-}\phi$ (sometimes in the past ϕ), $\mathsf{G}^{+}\phi$ (always in the future ϕ), $\mathsf{G}^{-}\phi$ (always in the past ϕ), $\phi\ \mathsf{W}^{+}\psi$ (ϕ weak until ψ, that is, ϕ is always true in the future or until ψ), $\phi\ \mathsf{W}^{-}\psi$ (ϕ weak since ψ), $\phi\ \mathsf{Z}^{+}\psi$ (ϕ until ψ and then no longer ϕ). This last operator may need further elaboration, as to our knowledge it is not used elsewhere. It is defined as follows,

$$\phi\ \mathsf{Z}^{+}\psi =_{def} \phi\ \mathsf{W}^{+}\psi \wedge \mathsf{G}^{+}(\psi \rightarrow \mathsf{G}^{+}\neg\phi),$$

which holds iff ψ never becomes true and ϕ is always true, or eventually ψ becomes true and from that moment on ϕ is no longer true.

In our view, events are reified, and each event token belongs to at least an event type, and takes place at exactly one instant. An action is an event brought about by an agent, called the *actor* of the action. We write $Done(e, x, \tau)$ to mean that event e of type τ is brought about by agent x. We use the "m-dash" character as a shorthand for existential quantification. For instance:

$$Done(e, -, \tau) =_{def} \exists x Done(e, x, \tau).$$

In our view, a commitment is a social state between agents comprised of four components: an *event* e that has created the commitment, a *debtor* x which is the agent who is committed, a *creditor* y which is the agent the debtor is committed to, and a *content* u which represents the state of affairs the debtor is committed to bring about, as in the following predicate:

$Comm(e, x, y, u)$.

As we do not want to depart from first-order logic, the content is represented by a term u, and we write $\lfloor u \rfloor$ to refer to the relevant LTL$^{\pm}$ formula.

Intuitively, a commitment is fulfilled when its content is true, and is violated when its content is false:

$Fulf(e, x, y, u) =_{def} Comm(e, x, y, u) \wedge \lfloor u \rfloor$,
$Viol(e, x, y, u) =_{def} Comm(e, x, y, u) \wedge \neg \lfloor u \rfloor$.

Agents create and cancel commitments by performing suitable tokens of *commitment manipulation* action types, like make commitment (mc) and cancel commitment (cc). The reader may refer to [15] for further details about commitment manipulation. The effects of the performance of a commitment manipulation action are illustrated in the form of axioms that reduce the number of allowable models by limiting the interpretation of our primitive predicates. The scope of the validity of formulae is then limited to the class of LTL$^{\pm}$ models that fulfill the constraints imposed by such axioms. For instance, if an agent (not necessarily x or y) performs an action of making a commitment with x as debtor, y as creditor, and u as content, then the relevant commitment holds until it is either cancelled, or fulfilled, or violated, after which it no longer exists:

$$Done(e, -, mc(x, y, u)) \rightarrow Comm(e, x, y, u)\mathsf{Z}^{+} (Done(-, -, cc(e, x, y, u)) \vee Fulf(e, x, y, u) \vee Viol(e, x, y, u)). \quad \text{(MC)}$$

We suppose that the multiagent system has been implemented in accordance with the commitment-related axioms, while our aim is to check whether some other properties, depending on actions performed by the agents, hold. Let us take φ as an example of an LTL$^{\pm}$ formula which we want to monitor at runtime:

$\varphi = \mathsf{G}^{+}(Comm(-, -, -, u_1) \rightarrow u_1)$, where $\lfloor u_1 \rfloor = \mathsf{X}^{+}(\mathsf{X}^{+}A)$.

This formula will be taken as an example in the reminder of the paper.

3 The Monitoring System

We may assume that every agent in the system can monitor its properties thanks to two specific functional components it is provided with: the *Collector* and the *Formula Analyzer*.

The Collector collects information while conversations are carried out by the agent with the others. The collected information consists of the atomic propositions contained in the monitored formula. We treat commitment predicates as atomic propositions, by introducing an atom for each different commitment. We deal with primitive predicates in the same way, and treat the derivated predicates as compounds of the atoms corresponding to their primitive components. Thus, in our example φ, the collected atomic propositions are $comm\_u_1$ and A. The collected atoms are time-stamped using the system's clock, which every agent is assumed to have access to. The pool of collected data is potentially infinite in

size, but in the following we will show that the nature of LTL$^{\pm}$ can ensure that a finite memory suffices.

The Formula Analyzer can monitor the truth value of any LTL$^{\pm}$ formula at runtime, hence, it has to deal with all the temporal operators of this language, including the future-tense ones. In the monitoring process, the evaluation of such sentences depends step by step on the evolution of the multiagent system. When monitoring φ, we want to focus only on those parts of the system that affect the truth value of the atoms in φ. The Collector builds a sequence of atoms included in φ which are increasingly time-stamped. The sequence begins with the truth values of all the atoms in φ at the initial state. In the following, we call such a sequence a *word*. Hence, if a formula that contains future operators needs to be evaluated at present time, its truth value depends on the suffix of the word that will be created at runtime from that moment on.

The communication mechanism between the Collector and the Formula Analyzer is based on a simple Publish and Subscribe paradigm [3]. The Formula Analyzer can get the relevant data in two different ways. It can either ask the Collector for the new truth value of an atom everytime it changes (*change subscription*), or ask also for a periodical truth value check (*periodic subscription*), so that the Formula Analyzer is notified by the Collector everytime the truth value of the monitored atom changes, or one period passes by. The type of subscription depends on the temporal operators the atoms are contained in. If the atom appears only in an atemporal subformula or in an unbounded (i.e. U^+, U^-, F^+, F^-, G^+, G^-, W^+, W^- and Z^+) temporal operator, the Formula Analyzer makes a change subscription. If the atom is in a bounded operator (i.e. X^+, X^-), the subscription must be periodic. Notice that if an atom is contained in more than a subformula and it calls for more than one type of subscription, the periodic subscription overrides the change one.

Let us consider again formula φ. It contains the atomic proposition A and the commitment associated with u_1, hence the Analyzer has to subscribe to A, using a periodic subscription, since A is contained in the two nested bounded operators X^+, and to $comm\_u_1$ with a change subscription. Let us suppose the monitored system evolves as in Figure 1 and consider instant 7 as present time (0 is when the relevant conversation started). We take the period to be one second long and suppose that the sequence of states in our model is finely graned enough to allow us to associate any event happening between two subsequent seconds to a

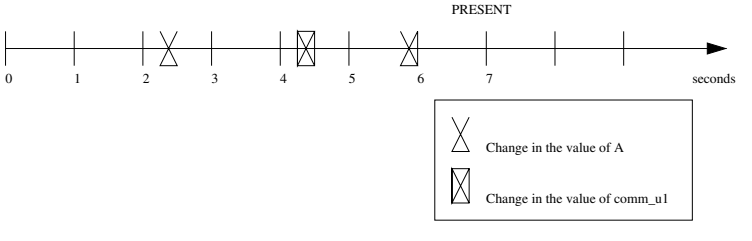


Fig. 1. A possible evolution of the system with respect to formula φ. Initially A is true and $comm\_u_1$ is false.

particular state. The atoms collected until the present instant, i.e. the prefix of the analyzed word, are:

$$(\{A, comm\_u_1\}, 0)(A, 1)(A, 2)(A, 2.4)(A, 3)(A, 4)(comm\_u_1, 4.3)(A, 5)(A, 5.7)(A.6)$$

and $(A, 7)$ is the current letter of the word. The Collector composes the collected atoms with the latest truth values of all the other atoms included in the monitored formula, as follows:

$(\{A, comm\_u_1\}, 1)(\{A, comm\_u_1\}, 2)(\{A, comm\_u_1\}, 2.4)(\{A, comm\_u_1\}, 3)$
$(\{A, comm\_u_1\}, 4)(\{comm\_u_1, A\}, 4.3)(\{A, comm\_u_1\}, 5)(\{A, comm\_u_1\}, 5.7)$
$(\{A, comm\_u_1\}, 6)$

and $(\{A, comm\_u_1\}, 7)$ is the current letter. For instance, the collected letter $(A, 1)$ becomes $(\{A, comm\_u_1\}, 1)$, where $comm\_u_1$ takes the same value it had at 0, because no changes have been published in the meantime. The word so far has been composed using the history of the changes of A and $comm\_u_1$ and the value of A at every second. The evolution of the system in the future with respect to A and $comm\_u_1$ is seen as the suffix of the word that will be built at runtime while the system evolves.

In general, the Formula Analyzer deals with a set Φ of formulae. If $|\Phi| = n$, it launches n independent components that work in parallel, each one devoted to a formula $\phi_i \in \Phi$. Every component is implemented as a *finite state automaton* A_i that has to recognize if the input letter, i.e. the time-stamped atom provided by the Collector, satisfies ϕ_i. The automaton A_i is built considering the nature of the formula it represents. LTL$^{\pm}$ allows for both past and future modalities. Our basic technique to deal with both is to consider the two modalities separately, i.e. with no nesting between them. This does not impose any restriction on the syntax of LTL$^{\pm}$, because a well known theorem by Gabbay [10] states that it is always possible to separate past-directed and future-directed subformulae with a non-elementary blow up in the size of the initial formula. This complexity may result in a significant impact on the monitoring system's efficiency in worst case scenarios, but it has to be noticed that logic formulae of some interest in a multiagent system are generally already separated or have a rather small number of nestings of past and future-directed operators.

Thus, we can suppose to work with separated formulae, i.e. boolean combinations of pure past-directed and pure future-directed formulae, and we deal with them in two different ways.

For the future modalities we exploit the well-known correspondence between LTL and *alternating automata* [4]. Alternating automata are a more compact form of Büchi automata, i.e., non deterministic finite state automata over infinite words [14]. An alternating automaton can be intuitively described as follows. In a deterministic automaton, the transition function maps a $\langle$*state*, *input-symbol*$\rangle$ pair to a single state, called the *next-state*. A *configuration* is a pair composed of the current state and the yet unread input content. By reading the next input symbol, the automaton can reach–in a single step–a new configuration. At the beginning, the automaton accepts an input x if the configuration reached from the initial state through x is acceptable. On the other hand, in a non-deterministic automaton a $\langle$*state*, *input-symbol*$\rangle$ pair is mapped to a set of states, hence the

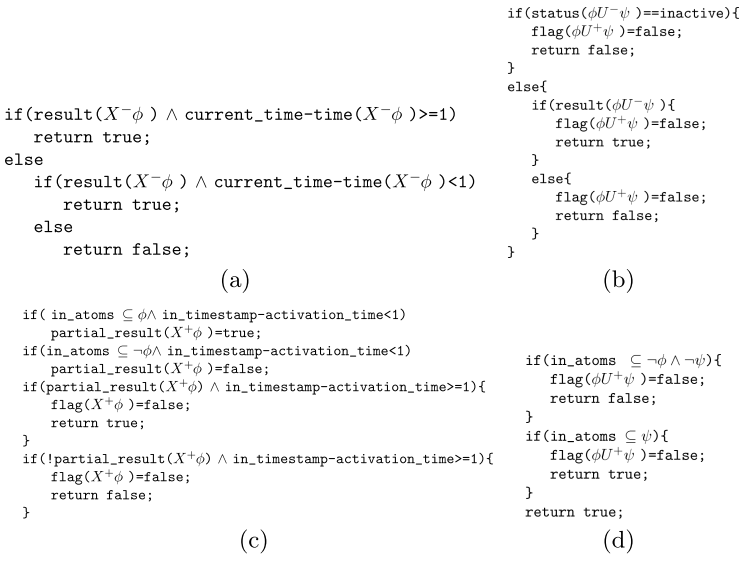


Fig. 2. Pseudo-code for monitoring (a) $\mathsf{X}^-\phi$, (b) $\psi\ \mathsf{U}^-\phi$, (c) $\mathsf{X}^+\phi$, and (d) $\psi\ \mathsf{U}^+\phi$. In the code in_atoms $\subseteq \phi$ is true if the current atoms value makes ϕ true; if ϕ is an atom, it is more correct to write in_atoms $\ni \phi$. Variable activation_time is a parameter that stores the value of the clock when a process is activated, while time($\mathsf{X}^-\phi$) stores the last time instant in which ϕ changed its value. Boolean variable partial_result($\mathsf{X}^+\phi$) saves the latest value of ϕ, stored before the one time unit is passed from the activation time, flag stores the status of the processes and result the present evaluation of past operators.

automaton may move, by reading the next input symbol, from a configuration to a set of configurations. The traditional interpretation of non-determinism is *existential branching*: an input x is accepted is there is at least a sequence of acceptable configurations which is reachable from the initial state. Another interpretation of nondeterminism is *universal branching*: a non-final configuration is acceptable if every configuration reachable in one step is acceptable. An alternating automaton provides both existential and universal branching. Its transition function maps a $\langle state, input\text{-}symbol \rangle$ pair into a (positive) boolean combination of states. Quite naturally, $\wedge$ is used to denote universality, while $\vee$ denotes existentiality.

For the past we do not need nondeterminism, as when a past-directed formula is evaluated at a certain state, the past is already set and the evaluation is immediate. However, we still need an automaton on infinite words since we intend to verify a past-directed formula in all time instants in the future, which makes us interpret the past as an infinite set of prefixes. In [11], it is shown that deterministic Büchi automata suffice for this purpose. The combination of different subformulae is performed easily, since the boolean connectors, $\wedge$ and $\vee$, naturally correspond to universal and existential branching, respectively,

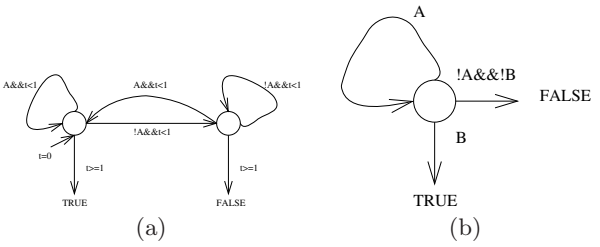


Fig. 3. Automata for the monitoring of (a) X^+A and (b) $A\ \mathsf{U}^+B$. && and ! are equivalent to $\wedge$ and $\neg$ respectively.

and deterministic Büchi automata can be seen as alternating automata without existential and universal branching.

The overall formula representation in the Formula Analyzer is inspired by this theoretical model, but with some limitations so to ensure that the alternating automata come in a finite number of copies and that a finite memory suffices for the Collector. The idea is that the automaton monitoring a formula is executed by a process, and all processes are coordinated by the Formula Analyzer, which receives the needed atoms from the Collector and passes them to the relevant processes. Thus, a generic Analyzer that deals with `N` processes executes the following loop:

```
while(true){
     in_atoms=new atoms;
     in_timestamp=new timestamp;
    (...)Sends the new collected data to the process waiting for truth values;
     for(int i=1; i<=N; i++)
         if(!result(i))
              break;
}
```

The variable `in_atoms` contains the latest value of all the atoms and `in_timestamp` is the corresponding time-stamp. The `atoms` and `timestamp` functions collect the latest atoms and time-stamp, respectively, from the Collector, while `result(i)` is a function evaluating the truth value of the formula monitored by process `i`. Notice that every time the automaton performs a universal branching, each new copy of the automaton is executed by a new independent process. In Figure 3 we propose an algorithmic representation only for X^+, X^-, U^+ and U^-, since all the other temporal operators can be derived by them. Moreover Figure 3 represents the automata corresponding to X^+ and U^+, while the automata for X^- and U^- are not reported since they trivially evaluate the value of the formula in a transition using the past values stored in the Collector, computed as in Figure 3. Notice that the communication between the Analyzer and a process is performed only when the process is active, i.e., when the corresponding formula needs to be evaluated. The other operators can be trivially derived by these four, however in the implementation of the monitoring system some optimizations in

```
if(in_atoms ⊆ φ∧previous==false){
   time(X⁻φ )=in_timestamp;
   result(X⁻φ )=true;
}
else
   if( in_atoms⊆ ¬φ∧previous==true){
       time(X⁻φ )=in_timestamp;
       result(X⁻φ )=false;
   }
```

(a)

```
if(in_atoms ⊆ ψ){
   status(φU⁻ψ )=active;
   result(φU⁻ψ )=true;
}
else
   if( status(φU⁻ψ )==active ∧ in_atoms ⊆ φ)
       result(φU⁻ψ )=true;
   else
       result(φU⁻ψ )=false;
```

(b)

Fig. 4. Process for pre-computing the value of the past operators (a) $\mathsf{X}^{-}\phi$ and (b) $\psi\ \mathsf{U}^{-}\phi$. The variable previous stores the last value assumed by ϕ.

the translation to automata are proposed to improve the overall performance. Moreover, nested X^{+} (X^{-}) operators are dealt as new operators, in which, instead of focusing on one time unit, we consider as many units as the number of nestings.

When the G^{+} and G^{-} operators are externally applied to the overall formula, they are not treated as temporal operators, but as universal quantifiers over time instants that are dealt with by continuously checking the quantified formula.

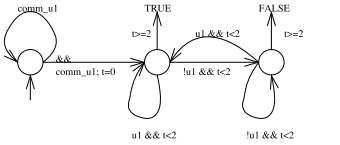
Notice that time-stamps offer us the possibility of a future extension in which metric temporal operators are introduced. Thus, since conversations are time consuming, we can guarantee that once we have fixed a time granularity, the number of data changes in a time unit is finite.

In an alternating automaton, every time a universal branch is taken multiple copies of the automaton are created to visit all the paths, which could theoretically lead to an infinite set of automata. However, in our implementation, the number of processes is always bounded by 1 for every operator.

In the example formula φ, we have a process that runs continuously to deal with the external universal quantification and for every time unit it checks whether $comm\_u_1$ holds and if so, it launches a new process devoted to u_1, while it goes on checking for new instances of $comm\_u_1$. The process that deals with u_1 checks two nested X^{+} operators, hence it saves the present time-stamp and waits for a change in the truth value of A. When this happens or the difference between the present time and the time-stamp becomes bigger than two time units, the process evaluates the formula as false, otherwise as true. If this last process evaluates the formula as false the overall value of the formula is false and it is notified to the Formula Analyzer. The pseudo-code and the automaton corresponding to the formula are represented in Figure 3.

4 Related and Future Work

The idea of having a separated module for system monitoring has already been proposed in the Web Service literature [1,2], which has inspired us for this work. To our knowledge, our automata-based monitoring approach is rather new in the context of multiagent systems, although some research about monitoring agents' social states has already been carried out. The most relevant related



(a)

```
Process φ
if( in_atoms ⊆ comm_u1){
    activate Process(X+ a, current_time);

Process(X+ a, activation_time)
if( in_atoms ⊆ a ∧ in_timestamp-activation_time<1)
    partial_result(X+φ )=true;
if(in_atoms ⊆ ¬ a ∧ in_timestamp-activation_time<1)
    partial_result(X+φ )=false;
if(partial_result(X+ a ∧ in_timestamp-activation_time>=1){
    flag(X+ a)=false;
    return true;
}
if(!partial_result(X+ a ∧ in_timestamp-activation_time>=1){
    flag(X+ a)=false;
    return false;
}
```

(b)

Fig. 5. Automaton (a) and correspondent processes (b) to monitor the example formula φ. Process φ activates the process corresponding to $\mathsf{X}^+\ \mathsf{X}^+$ setting its flag to true and from that moment on the two processes communicate every time the Analyzer sends a new letter to process φ until the flag of $\mathsf{X}^+\mathsf{X}^+$ is set to false again.

work is Cranefield's [7], in which hyMITL$^{\pm}$, a rule language for specifying social expectations, is presented and an algorithm for rule compliance monitoring is outlined. hyMITL$^{\pm}$ relies on a branching model and is more expressive, which makes the monitoring process more complex a task. The current status of our work calls for a detailed comparison of the two approaches to establish the correct balance between expressiveness and efficiency. In particular, we hope that our automata-based operational approach will help tackle some of the technical issues in hyMITL$^{\pm}$ related to the evaluation of the formulae's truth value. Since our approach is based on the evaluation of time-stamped words, it can be naturally extended to deal with more complex metric operators and functions and work with dense time models.

References

1. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-Time Monitoring of Instances and Classes of Web Service Compositions. In: ICWS'06. Proceedings of the IEEE International Conference on Web Services, pp. 63–71. IEEE Computer Society Press, Los Alamitos (2006)
2. Baresi, L., Guinea, S.: Towards Dynamic Monitoring of WS-BPEL Processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 269–282. Springer, Heidelberg (2005)
3. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems 19(3), 332–383 (2001)
4. Chandra, A.K., Kozen, D., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
5. Colombetti, M.: A Commitment–Based Approach to Agent Speech Acts and Conversations. In: Agents 2000, Barcelona, Spain, pp. 21–29 (2000)
6. Colombetti, M., Verdicchio, M.: An Analysis of Agent Speech Acts as Institutional Actions. In: Castelfranchi, C., Lewis Johnson, W. (eds.) AAMAS 02. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1157–1166. ACM Press, New York (2002)

7. Cranefield, S.: A Rule Language for Modelling and Monitoring Social Expectations in Multi-Agent Systems. In: Pack Kaelbling, L., Saffiotti, A. (eds.) IJCAI'05. Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, pp. 1659–1660 (2005)
8. Emerson, E.A.: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. MIT Press, Cambridge, MA (1990)
9. FIPA: ACL Specifications. Specification, Foundation for Intelligent Physical Agents (2002), `http://www.fipa.org/repository/aclspecs.html`
10. Gabbay, D.M.: The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 409–448. Springer, Heidelberg (1989)
11. Pradella, M., San Pietro, P., Spoletini, P., Morzenti, A.: Practical Model Checking of LTL with Past. In: ATVA 2003. Proceedings of the 1st International Workshop on Automated Technology for Verification and Analysis, National Taiwan University (2003)
12. Reynolds, M.: More Past Glories. In: Proceedings of the Fifteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2000), pp. 229–240. IEEE Computer Society Press, Los Alamitos (2000)
13. Singh, M.P.: Agent Communication Languages: Rethinking the principles. IEEE Computer 31, 40–47 (1998)
14. Thomas, W.: Automata on Infinite Objects. In: Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics, pp. 133–191 (1990)
15. Verdicchio, M., Colombetti, M.: A Logical Model of Social Commitment for Agent Communication. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, Springer, Heidelberg (2004)

Competencies and Profiles Management for Virtual Organizations Creation

Jiří Hodík[1], Jiří Vokřínek[1], Jiří Bíba[1], and Petr Bečvář[2]

[1] Gerstner Laboratory, Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague, Czech Republic
{hodik|vokrinek|biba}@labe.felk.cvut.cz
[2] CERTICON Corp., Applied Research, Czech Republic
becvar@certicon.cz

Abstract. Sharing information about profiles and offered competencies of individual members within an alliance of cooperating companies facilitates searching for potential members of Virtual Organizations. This paper presents a concept of structuring of the competencies as well as a prototype proving this concept. The prototype follows a naturally hybrid architecture of alliances that consists of mutually independent alliance partners, who may be supported by central institutions of the alliance. The use of the agent-based solution enables information sharing among partners in such distributed and dynamic environment.

1 Introduction

In business-to-business (B2B) e-commerce, a group of collaborating partners may act as a single company and create a more competitive whole [14] in order to exploit businesses that cannot be executed by individual companies on their own. The work of Davidow and Malone from early 90's suggest Virtual Corporations as an industrial strategy for the 21st century [4]. Though innovative, this approach introduces a logical continuation of existing collaboration strategies [17]. During the last years, the concept of Virtual Corporations has evolved into various collaboration models. In our work we focus on the original concept of Virtual Organizations (VO). The results may be applied to the other collaboration models as well (e.g. Extended Enterprise).

Gruber specifies the key features of VO also defined by most of another definitions as: *an extensive use of information technology for a coordination of the partners*, *sharing risk and knowledge with partners*, and *focus on core competencies* [9]. Defining a *competence* Neubert refers to *"the cognitive, conative and expressive abilities of humans to organise their activities in order to produce certain results"* [13]. Neubert suppose the competence to be a necessary prerequisite *"realizing a business process to create valuable results"* [13] for each VO member. Fischer describes a *core competence* of an enterprise as a set of skills, technologies, and know-how crucial for the added value provided by the enterprise [6].

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 93–102, 2007.

This paper presents a concept of profiles and competencies management for alliances in B2B domain. The goals of our work are simplification and integration of cooperation between the partners facilitated by means of sharing information about their competencies. The concept is based on a hybrid architecture consisting of peer-to-peer cooperating units supported by centralized components. Such central components are inherited from the architecture of alliances (e.g. clusters or Virtual Organization Breeding Environments) that are formed by companies in order to facilitate formations of Virtual Organizations. The alliances may support their members also by means of tools for effective partner search, social knowledge management, negotiation support and others. The concept presented in this paper is proved on a prototype tool e-Cat, which is build upon a distributed (multi-agent) framework.

2 Theoretical Background

Adesta explains that information and communication systems are crucial for effective collaboration. Referring to an other work, Adesta calls information to be the blood of Virtual Organizations [1]. In this section we focus on distributed architectures for information sharing, mainly on multi-agent systems that the framework for presented prototype builds upon.

2.1 Distributed Architectures for Information Sharing

A common solution used for document management in teams consists in a central point maintaining the shared documents and enabling an access for all team members. Friese compares features of centralized (client-server architecture) and distributed technologies for information management. Without a central server, bottlenecks and single points of failure are avoided and individual peers keep their independence. On the other hand, network administration lacks any central control, which is required in some domains to ensure consistency and verity of data within the network [7]. Jun Yan *et Al.* also discuss weaknesses (mainly architectural limitations) of conventional workflow management systems: poor performance, lack of reliability, limited scalability, user restriction, and unsatisfactory system openness [18]. Examples of distributed architectures for information sharing during the VO creation process and further workflow management tasks are: peer-to-peer networks, grids, and multi-agent systems.

The peer-to-peer (P2P) network is a distributed technology consisting of mutually independent entities (peers) having respective physical and logical resources [7]. The only effort required for a peer to connect to a P2P network is to implement the corresponding network protocol [16]. In order to distinguish between different kinds of networks, Friese defines (beside the centralized architecture) a pure P2P network, consisting of "equal" peers, and a hybrid P2P network, which contains specialized nodes for certain functions. Friese presents a Resource Management Framework used in a P2P business resource management framework for managing resources such as web services and business process execution engines [8]. The P2P architecture may be also utilized for decentralized

workflow management system of VO (e.g. SwinDeW by Jun Yan *et Al.* [18]). The maintenance of distributed and heterogeneous knowledge bases is widely studied also in works related to the Grid technology.

Another example of a distributed architecture is the multi-agent technology (more detailed explanation of the technology is in the next section). The KRAFT architecture presented by Preece supports VO by "knowledge fusion" [14]. Referring to earlier works Preece specifies two essential common mechanisms of successful integration to VO: an exchange electronic documents, and a synchronization of workflows. The KRAFT employs the multi-agent technology to integrate all local data sources and other knowledge-processing components. Limitation of the KRAFT is its orientation only to pure cooperative environments. Another agent-based system is MAIQS by Linn, which concentrates on cooperative document indexing and querying in geographically distributed networked environments of 5–20 workers collaborating in peer-to-peer manner [12]. Agents support their users by searching for and indexing of workgroup data and documents. In the MAIQS, there is no centralized data store, register, or broker; all information is stored on workstations of the team members and its exchange is queried by the agents directly. The membership in the workgroup is resolved by means of configuring a respective agent to be aware of the other members in the workgroup as well as by means of configuring the other agents in the workgroup to be aware of the respective agent.

2.2 Multi-agent Systems

Multi-agent technology provides mechanisms for information exchange, negotiation, and team action coordination. These features are very suitable in VOs and their workflows management [14].

A multi-agent system is a technology of distributed artificial intelligence. This technology is used to link logically or geographically distributed systems together or to model negotiation in such systems. Different requirements may lead to different agents. For business environment the agent "is an agent of someone" and plays any of the following roles:

- **Assistant.** Agent's task is to acquire and analyze information in order to support the human operator's (partner's) decision making.
- **Representative.** It is a tool used for collaboration with others. Negotiation with an agent within the scope of its power is equivalent to negotiation with the partner.
- **Model.** A simplified representation of an original (real) partner equipped with required attributes and abilities sufficiently consonant to attributes and abilities of the original partner. For defined domain the agent is a model of the partner and may act on behalf of the partner by means of emulating the partner's behavior.

Agents are allowed to share their knowledge and dynamically form teams to achieve their goals [5]. Such goals do not need to be necessarily related or compatible [2]. An agent also keeps knowledge about other agents as well as

the level of confidence in that knowledge. In the area of multi-agent systems the concept of clustering individuals into cooperating groups is often used (e.g. Pěchouček in [15]). Pěchouček defines an *alliance* and a *coalition*. The former one (alliance) is a collection of cooperating units that share semi-private knowledge (e.g. information about resources if considered as non-private knowledge) and all agree to form possible coalitions. The alliance is regarded as a long-term collaboration agreement among the units. The latter one (coalition) is a set of units agreed to collaborate to fulfill a single, well-specified goal. A coalition, unlike an alliance, is thus usually regarded as a short-term agreement between collaborating agents.

3 Competency Terminology

Since there is no common terminology in competency management domain, the terms "competency", "competency class", "competency instance", and "profile" are used in several slightly different meanings. In some cases, the term competency is used only for a competency class (e.g. Biesalski [3] or NASAPeople[1]). On the other hand, e.g. HR-XML[2] (focussing mainly on human resources management) uses the term "competency" both for the competency class and the competency instance. Here we suggest definitions of the competency related terms. These definitions are based on our previous work in this domain [11]. As the first step we define the *competency* and the *profile*:

- **Competency** is *"an ability to perform business processes, which are supported by necessary available resources, practices and activities, allowing the organization to offer products/services."*
- **Profile of subject** contains *"two main elements: (i) general information about the partner, and (ii) a set of competencies offered by the partner as their core competencies"*

The competencies presented in the profiles are derived from commonly understood and accepted definitions in order to ensure consistency and avoid misinterpretation of the presented competencies. Then we distinguish between the *competency class* and the *competency instance*:

- **Competency class** defines *"an existence of the competency in the world; it distinguishes it from other existing competencies and defines relations to them. Competency class may be extended by defining means used for measuring the level and robustness of the competency.*
- **Competency instance** refers *"exactly to one competency class related to one subject[3]. If the competency class defines means for measuring level and*

[1] http://nasapeople.nasa.gov

[2] http://ns.hr-xml.org

[3] The relation between competency classes and their instances resembles the relation between classes and instances (objects) in object oriented programming. Each subject may instantiate several competency classes, while two instances of the same class instantiated by two different subjects differ.

robustness of the competency, the competency instance can optionally assign values to them."

As the competencies in the real world are more-or-less related, competency classes also provide for descriptions of such relations. There is no more than one direct relation between each two competencies. The relations of one competency to others are:

- **Specialization.** A competency does not need to be specific enough in some cases. Then, a specializing competency may exist to extend some features of the original competence.
- **Generalization.** Generalization introduces an opposite to specialization. A competency specialized by one or more competencies introduces a generalization of all those competencies.
- **Affinity (also "relation to").** A competency may be somehow related to another one, but the relation is neither generalizing nor specializing; the relation may be unspecified – either due to its complexity or due to its intuitiveness. Such relations are present only in some competency structures.
- **Closeness.** It describes similarity, relation or coherency between each two competencies. It enables search for alternative competency if desired one is not available (not instantiated anyone or competency owner is not available for whatever reason). It can be defined explicitly between every two competencies or defined as a metrics using specialization, generalization and affinity relations.

Allowed relations and their cardinalities are determined by rules that are defined by structure of competencies. The structures are described by graphs, where the competencies are defined as nodes and their relations as edges. The relation of generalization/specialization correspond to oriented edge and the relation of affinity (if present) to non-oriented edges. None of competency structures allows a cycle of oriented edges – thus none competency may be its own direct or indirect generalization/specification. The common structures according to ordering of the oriented edges are:

- **Hierarchical structure (oriented tree).** In such case there exists an only general competency that is on the top of others and introduces a common root of the all generalizing competencies. Each competency (except the top one) has defined exactly one generalizing competency while it may have defined several or none specializing competency.
- **Heterarchical structure (oriented forest).** It is similar to the hierarchical structure, but there may exist several roots. There may exist competencies without any competency generalizing them.
- **Multiarchical structure (oriented acyclic graph).** There is no limitation for number of any relation type for any competency. Each competency may have unlimited number of generalizing and specializing relations to other competencies.

The type of a structure that competencies are organized in has a strong impact on the competency search mechanism. For hierarchical structures there is a

relatively simple and explicit way of defining closeness of the competencies (e.g. number of edges between two competencies). For heterarchical structures there may defined an explicit non-parametric evaluation of closeness for competencies belonging to two disjoint trees while for multiarchical structures even more complex metrics of closeness may be defined.

4 The e-Cat System

The e-Cat is an agent-based research prototype of a tool for the facilitation of members' profiles and competencies in alliances of SMEs. e-Cat aims at the identification of potential partners; it is used for keeping, managing and distributing profiles of alliance members. The presented technology is based on a distributed set of agents, representing individual members, which are supported by centralized elements. Such a hybrid peer-to-peer network architecture (defined by Friese [7]) enables effective cooperation in a distributed environment where agents ensure maximal independence between alliance members and private knowledge preservation.

To be found as a potential partner, an SME provides other alliance members with its profile containing identification, contact information and competencies that the SME offers as its core competencies. Each competency (instance) is instantiated from a competency class, which is known to all alliance members. Each agent stores description of core competencies on a member's local server and maintains it individually. Access to information is based on roles – only the responsible agents are authorized to edit it, the others may only read them (Only if they are authorized. This feature is required for information privacy.) If some information is expected to be out of date, the corresponding partner or information source is asked for an update of the information (agents are able to use subscribe-advertise or query protocol when negotiating the information). Identification and contact information of alliance partners as well as definitions of competency classes are maintained by alliance institutions. Centralized alliance services support alliance members by: (i) ensuring the common understanding of competencies and understanding member's profiles within the whole alliance, and (ii) maintaining identification information of alliance members (centralized part of member's profile) to limit the access to the community only to the authorized members and to prevent pretending to act as another company. Members' agents download information from central elements when needed, or they store a local copy of the centrally maintained information (or its appropriate part). The figure 1 presents use cases for e-Cat. The copy of frequently accessed information allows each member to use the tool, even if it is totally disconnected from the rest of the world. The e-Cat consists of following subsystems:

- **Distributed Profile Catalogue** keeps, manages and distributes members' profiles. Each member is responsible for its distributed part of the profile. A member has read-write access to its profile and read-only access to other profiles.

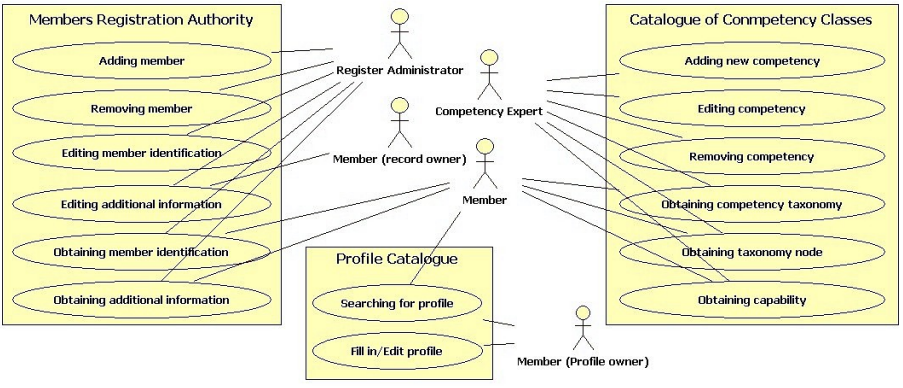


Fig. 1. Use cases of e-Cat

- **Catalogue of Competency Classes** defines the competencies that can be advertised through e-Cat, their exact descriptions, taxonomy and attributes. It ensures coherence in the common schema of competency. The Catalogue of Competency Classes is maintained by a competency expert of the alliance, who is responsible for information in the catalogue. Only this expert is allowed to edit the catalogue; the others may read it and suggest improvements.
- **Members Registration Authority** allows full control of members entering or leaving an alliance, and maintains information for member identification. This part is designed as a centralized element to allow the alliance management to control the members entering the community. The Members Registration Authority is under control of subject (e.g. VBE manager) entrusted by the alliance.

The taxonomy of competencies in the e-Cat is represented by a common graph and the closeness of competencies is defined as shortest path in the graph (for the path count all edges in the graph are considered to be non-oriented with constant weight).

Members Registration Authority and Catalogue of Competency Classes are intended to be deployed on alliance management servers and maintained by responsible experts of alliance support institutions. The Profile Catalogue is naturally distributed and it may be deployed on members' servers. Members can also share servers (the location of physical equipment is irrelevant) to install their parts of the Profile Catalogue. A specialized part of the Profile Catalogue may be instantiated on a server of the alliance management to summarize information and represent the profile of the alliance as a whole.

The e-Cat prototype is distributed system implemented in JAVA language using JADE[4] (Java Agent Development Framework) empowered by FIPA[5] in-

[4] http://jade.tilab.it/
[5] http://www.fipa.org/

teraction protocols. The agent negotiation is based on concept of enterprise-to-enterprise (E2E) agents [10] and uses Apache Jakarta Tomcat[6] powered web-based user interface. The e-Cat implementation details can be found in [11] too.

4.1 Usage Scenario

Let us assume a hypothetical, but realistic, scenario of the e-Cat usage. The company "Dirk Gently" is a SME, specialized in transport and sale services. The company is willing to join an alliance. The company contact information is added to the e-Cat and the company competencies are put into the profile to be available to other members. Thanks to e-Cat, a member looking for services offered by the "Dirk Gently" company is able to find it and contact the company.

- ***Joining the e-Cat community and creating a new profile.*** As a first step the "Dirk Gently" installs e-Cat on the server. The following configuration includes adding the addresses of Members Registration Authority and Catalogue of Competency Classes. These addresses are provided to the company during the process of joining the alliance. After that, the company can be included into the register of members and thus to be added to the community. At this moment, the expert of the Members Registration Authority creates a new record in the register and adds basic member's contact information to it. The record also contains the name and address of the new e-Cat component representing the new member.
- ***Announcing a competency.*** If Dirk Gently decides to offer some services to other alliance members, the competency class for such services is instantiated in its profile. The appropriate competency class is found in Catalogue of Competency Classes. If the proper class does not exist in the catalogue, either a generalizing competency is used or the catalogue expert adds new class to the catalogue and then the class is used.
- ***Creating a new competency.*** When a request for adding a new competency class to the catalogue appears, the competency expert reviews the request and decides whether to accept it and adapt the catalogue or not. e-Cat users are automatically notified once the catalogue is updated.
- ***Looking for a provider of a competency.*** The search engine of the e-Cat information system offers various attributes for finding potential partners among e-Cat users. The local copy of profiles of other members is searched for the requested competency. If the local copy of profiles is lost or outdated, partners are asked for information dynamically. If the search result is unsatisfactory, user can decide to use the taxonomy to find a generalizing or specializing competency and search profiles for them. In fact, the first search looks for available partners with instantiated competencies of closeness to searched competency equal to zero. Next, the closeness constrain is relaxed until a suitable partner is found.

[6] http://jakarta.apache.org/tomcat/

5 Conclusion

The presented system e-Cat focuses on advertising SME's competencies and searching for potential business partners to gain a competitive advantage (in order to exploit business opportunities which a single SME cannot cover on its own). The e-Cat architecture utilizes hybrid P2P networks for profile maintenance and sharing within an alliance of SMEs. The e-Cat has been verified using three different alliance datasets. The first dataset has been acquired from Virtuelle Fabrik[7] and used for a verification of the architecture on real data. The second dataset has extended the first one and has been used for a presentation of all features of the system. The third dataset has been provided by an alliance organized by IECOS[8] that had evaluated e-Cat in daily use.

The e-Cat architecture has been evaluated for independent alliances employing own supporting institutions. Possible future extensions of the presented concept are: (i) a connection to partner's ERP systems for automated negotiations on available capacities and contract details; (ii) a connection to alliance reputation management systems in order to make accessible information about the alliance members' performance in already created VOs; and (iii) tools for automated service level agreement negotiations.

Acknowledgement

The work is (part-)funded by the European Commission's FP6 programme within the projects ECOLEAD (contract NO. FP6 IP 506958) and PANDA (contract NO. 027169).

Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

References

1. Adesta, E.: A strategic planning procedure to support progress towards extended enterprise. In: ICE-2005. Proc. of the Eleventh Int. Conf. on Information and Computation Economies, Centre for Concurrent Enterprising, pp. 371–377 (2005)
2. Alan, L.G., Bond, H. (eds.): Readings in Distributed Artificial Intelligence. Morgan Kaufmann, San Mateo, CA (1988)
3. Biesalski, E.: Knowledge management and e-human resource management. In: FGWM 2003, Karlsruhe (2003)
4. Davidow, W., Malone, M.S.: The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century. Harper Business, New York (1992)
5. Farhoodi, F., Graham, I.: Practical approach to designing and building intelligent software agents. In: Proceedings of PAAM'96, pp. 181–204. The Practical Application Company (1996)

[7] http://www.virtuelle-fabrik.com/
[8] http://www.iecos.com/

6. Fischer, K., Heimig, I., Müller, J.P.: Intelligent agents in virtual enterprises. In: Proceedings of PAAM'96 (1996)
7. Friese, T., Freisleben, B., Rusitschka, S., Southall, A.: A framework for resource management in peer-to-peer networks. In: Proceedings of NetObjectDays 2002, pp. 4–21 (2002)
8. Friese, T., Müller, J.P., Smith, M., Freisleben, B.: A robust business resource management framework based on a peer-to-peer infrastructure. In: CEC 2005. Proceedings of 7th IEEE International Conference on E-Commerce Technology, pp. 215–222. IEEE Computer Society Press, Los Alamitos (2005)
9. Gruber, M., Nöster, M.: Investigating structural settings of virtual organizations. In: ICE-2005. Proc. of the Eleventh Int. Conf. on Information and Computation Economies, pp. 245–252. Centre for Concurrent Enterprising (2005)
10. Hodík, J., Bečvář, P., Pěchouček, M., Vokřínek, J., Pospíšil, J.: Explantech and extraplant: multi-agent technology for production planning, simulation and extra-enterprise collaboration. International Journal of Computer Systems Science and Engineering 20(5), 357–367 (2005)
11. Hodík, J., Bečvář, P., Vokřínek, J., Bíba, J., Semsch, E.: e-Cat – VBE members profiling and competency management tool. In: Intelligent Production Machines and Systems 2nd I*PROMS Virtual International Conference, July 3-14, 2006, pp. 126–131. Elsevier, Amsterdam (2006)
12. Linn, C.: A multi-agent system for cooperative document indexing and querying in distributed networked environments. In: ICPP Workshops, pp. 400–405 (1999)
13. Neubert, R., Langer, O., Görlitz, O., Benn, W.: Virtual enterprises – challenges from a database perspective. In: Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises, pp. 98–106. IEEE Computer Society Press, Los Alamitos (2001)
14. Preece, A., Hui, K., Gray, P.: Kraft: Supporting virtual organisations through knowledge fusion. In: Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop, pp. 33–38. AAAI Press, Stanford, California, USA (1999)
15. Pěchouček, M., Mařík, V., Bárta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems 17(3), 17–25 (2002)
16. Schmees, M.: Distributed digital commerce. In: Proceedings of the 5th International Conference on Electronic Commerce (ICEC 2003), pp. 131–137 (2003)
17. Walters, D.: Virtual organisations: new lamps for old. Management Decision 38(6), 420–436 (2000)
18. Yan, J., Yang, Y., Raikundalia, G.K.: SwinDeW-a p2p-based decentralized workflow management system. IEEE Transactions on Systems, Man and Cybernetics, Part A 36(5), 922–935 (2006)

Complexity of Verifying Game Equilibria

Emmanuel M. Tadjouddine

Department of Computing Science, King's College,
University of Aberdeen, Aberdeen AB24 3UE, Scotland
etadjoud@csd.abdn.ac.uk

Abstract. We consider the problem of verifying game equilibria in multi-agent systems. We first identify a certain class of games where Nash or Bayesian Nash equilibria can be verified in polynomial time. Second, we show that verifying a dominant strategy equilibrium is NP-complete even for normal form games. Eventually, we consider general games and discuss the complexity of equilibrium verification.

1 Introduction

As agent-based systems are increasingly used to model real-life applications, e.g., the Internet, electronic markets or disaster management scenarios, it is important to study the theoretical complexity of such usually combinatorial systems with respect to some desired properties. An example of such a study in the context of the emergence theory is presented in [1]. Moreover, an important niche of multi-agent systems is in the economics aspects of the Internet where the system can be modelled as a game and the resulting model is used to find out or prove some game-theoretic properties of the system, see for instance [2] for an enlightening discussion on the subject. This leads us to the fundamental results in game theory among them the existence theorem due to Nash [3] proving that non-cooperative games have a mixed strategy Nash equilibrium.

Recently, it is been shown that computing a Nash equilibrium for a normal form game is PPAD-complete for more than four players [4], three players [5,6] and two players [7]. PPAD (Polynomial Parity Arguments Directed version) [8] is the complexity class of NP problems that always have a solution and have the same existence proof. Such existence proofs can be carried out with this sort of argument: "if a finite directed graph has an unbalanced node, then it must have another one". In fact, PPAD is a subclass of the class NP. While the proofs of computing Nash for three, four or more players in [5,6,4] rely on colouring techniques on the graphical model of the normal form game, the 2-Nash's proof is settled by a reduction from 3-Nash [7].

Let us comment a bit on the importance of this complexity result on computing Nash equilibrium. An early example of PPAD-complete problem is the computation of Brouwer's fixed point theorem [8]. Although Nash's existence proof uses Kakutani's fixed point theorem, which is an extension of Brouwer's from functions to correspondences, computing a Nash equilibrium could be easier. However, this PPAD-completeness of computing a Nash equilibrium establishes

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 103–112, 2007.

a kind of computational equivalence between computing a Brouwer's fixed point and a Nash equilibrium since Nash's existence proof relies on Brouwer's fixed point theorem whose computation is PPAD-complete.

In this paper, we are interested in the verification of a given equilibrium point of the game. We assume the game as well as one of its equilibrium point are published in a manner that they become common knowledge to rational agents. The importance of achieving common knowledge in the system is discussed for example in [9]. Under these assumptions, we investigate the theoretical complexity of verifying that a given point is indeed an equilibrium for the game. This verification problem has applications in eCommerce wherein software agents (owned by humans) may compete for goods, move between auction houses to look for the best deal available, and be engaged in financial transactions just like humans do. For that purpose, they need to understand and trust previously unseen protocols. This is best illustrated by the automated mechanism design paradigm [10] wherein in a two-stage game the auctioneer may choose a particular protocol to satisfy a desired objective of the system. In such a scenario, each participant may be recommended an optimal strategy given its preferences. Should the participants that are rational trust their recommended strategies? This is not obvious since agents may not trust the mechanism designer. For example, an auctioneer may be driven by maximising its revenue at all costs even by telling lies. Furthermore, the protocol may be biased to suit some particular interests. It turns out the verification is crucial since it ensures that it is in the participant's interest to follow the recommended strategy, see [9,11] for more details.

Related Work. In [12], it is argued that the triple KRA (Knowledge, Rationality, Action) is a powerful tool for reasoning on multi-agent systems. Among other examples, the idea of checking equilibria for normal form games with complete information by using the KRA triple and the backward induction technique [13, p. 58] is presented. In [14], a WHILE-language is extended to represent game theory mechanisms with complete information for multiple agents and a Hoare-like calculus with pre- and post-conditions is used to verify simple mechanisms such as the Solomon's dilemma or the Dutch auction. In [15], the SPL (Simple Programming Language) [16] is extended into SMPL (Simple Mechanism Programming Language) in order to represent games for multi-agent systems where agents have implicit preferences. The equilibrium is checked using an exhaustive algorithm that builds up the entire game tree and uses a backward induction procedure to check equilibrium properties at each tree node. In [17], the game is constrained so as it can be represented as a set of Presburger formulae along with the equilibrium property, which is checked using a fixed-point approach. In [11], strategyproofness of an auction mechanism is verified using the Alloy model checker based on first order relational logic [18,19]. The hereby cited work did not address the complexity issues of carrying out an equilibrium verification.

The contributions of this paper are to have (i) showed that verifying dominant strategy equilibrium is NP-complete even for normal form games and (ii) identified certain forms of games and equilibria for which the verification is tractable. The remainder of this paper is organised as follows. Section 2 gives definitions

on equilibria of games and different representations of games, in particular the Graphical Game and the General Graphical Game representations. Section 3 is devoted to the verification of equilibria for normal form games. Section 4 investigates equilibrium verification of General Graphical Games. Section 5 concludes.

2 Definitions and Background

In mechanism design [20,10], the designer aims at achieving certain desired outcomes that induce certain strategies for the participants. The resulting mechanism may be *incentive compatible* (no agent can benefit from lying when its opponents are truthful) or *strategyproof* (no agent can benefit from lying regardless to what its opponents do). To achieve these mechanism properties, equilibrium strategies can be recommended to the agents according to their types or preferences. Each agent should make sure the recommended strategies are a true equilibrium of the game. But how easy or difficult this checking can be?

Let us consider a normal form game $\mathcal{N} = (N, (S_i, u_i)_{i \in N})$, in which

- $N = \{1, 2, \ldots, n\}$ is the set of n players
- S_i is the set of strategies for player i
- u_i is the utility for player i.

Assuming $(S_i)_{i \in N}$ has two elements, say $0, 1$, the $(u_i)_{i \in N}$ form a tabular matrix $M(\mathbf{x}), \mathbf{x} \in \{0,1\}^n$ containing 2^n n-tuples. This implies the game is represented by an exponential number of input data. When, the matrix M is given, the game is represented in *tabular* form. The actions 0 or 1 are called the *pure strategies* for each player.

Definition 1. *For a given normal form game* $\mathcal{N} = (N, (S_i, u_i)_{i \in N})$ *and a strategy profile* $\mathbf{s}^* = (s_1^*, s_2^*, \ldots, s_n^*)$, $\mathbf{s}^*$ *is a Nash equilibrium iff* $\forall i \in N, \forall s_i \in S_i, u_i(s_1^*, \ldots, s_{i-1}^*, s_i, s_{i+1}^*, \ldots, s_n^*) \leq u_i(s_1^*, s_i^*, \ldots, s_n^*)$.

If $(S_i)_{i \in N}$ are convex sets, then there exists a Nash equilibrium in the game. In other words, a Nash equilibrium does not always exist in pure strategy games. However, a way of making a set convex is to allow convex combinations over that set. This amounts to having a probability distribution over the set $\{0, 1\}$. For example, player i will play 0 with some probability $p_i \in [0, 1]$. This defines a *mixed* strategy for player i. Consequently, each pure strategy profile $\mathbf{x}$ corresponds to a mixed strategy $\mathbf{p}$ in which each x_j is 0 with some probability p_j and 1 with the probability $1 - p_j$. This leads to an expected utility matrix $M(\mathbf{p})$. Interestingly, in a mixed strategy game, there always exists a Nash equilibrium [3].

2.1 Graphical Games

To circumvent the difficulty of dealing with exponential input data required for game of interests such as auctions or electronic markets, wherein there may be many players and strategies, we should be able to have a compact representation of the game. For that purpose, Kearns, Littman, and Singh have introduced in [21], a graphical model called *graphical game* to represent such games.

Definition 2. *A graphical game is a data structure* $\{G, M\}$ *formed by*

- *an undirected graph* $G = (N, E)$ *wherein each node* $i \in N$ *represents a player and an edge* $(i, j) \in E$ *means player* i *influences the utility of player* j.
- *each node* $i, 1 \leq i \leq n$, *is associated to a local utility matrix* M_i *containing* 2^n *elements, which determines the utility of player* i. *The* M_i *form the set* M *of matrices.*

If the undirected edge $(i, j) \in E$, then j is a *neighbour* of i and vice versa. Let $N_G(i)$ be the set of neighbours of i (including i). If for all $i \in N$, $|N_G(i)|$ is bounded above by a constant number k, which is too small compared to n so that it makes economic to exploit the structure of the graph, the graph G is called *sparse*. In this case, the graphical representation is polynomial in the number of players and strategies while such a representation is exponential in general. An attractive property of the graphical representation is that it captures games made up of local interactions involving few players in a global situation. For example, the salesperson is involved in a local competition with salespeople in her neighbourhood. This local property gives rise to sparse graphs that can lead to computational savings when their structures are exploited. This graphical representation is generalised in [4] by making the graph $G = (N, E)$ directed. This clarifies the relationship expressed by an edge as an ordered pair $(i, j) \in E$ since it is antisymmetric, in essence. Player i can affect the utility of player j and the converse may not be true.

2.2 General Graphical Games

We further refined this graphical game representation as a directed graph to enable us dealing with games of incomplete information. These games have a normal form representation of the form $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$. Each agent i has extra features, which are its *type* space T_i and its *belief* p_i. Agent i's type, $t_i \in T_i$ is privately known by agent i and influences agent i's utility $u_i(t_i, s_1, s_2, \ldots, s_n)$. Agent i's belief $p_i(t_{-i}|t_i)$, describes i's uncertainty about the $n-1$ agent's possible types, t_{-i} given i's own type t_i.

Definition 3. *For a given normal form game* $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$ *and a strategy profile* $(s_1^*(t_1), \ldots, s_n^*(t_n))$, $(s_1^*(t_1), \ldots, s_n^*(t_n))$ *is a Bayesian Nash equilibrium iff* $\forall i \in N$ *and* $\forall t_i \in T_i$, $s_i^*(t_i)$ *solves the maximisation problem*

$$\max_{s_i \in S_i} \sum_{t_{-i} \in T_{-i}} u_i(s_1^*(t_1), \ldots, s_{i-1}^*(t_{i-1}), s_i, s_{i+1}^*(t_{i+1}), \ldots, s_n^*(t_n); t) p_i(t_{-i}|t_i). \quad (1)$$

In other words, a Bayesian Nash equilibrium is simply a Nash equilibrium in a Bayesian game. A stronger equilibrium property from the viewpoint of the game mechanism designer is strategyproofness. This is aimed at encouraging agents to be truthful and not to speculate about their opponents' types or preferences. In dominant strategy equilibrium, no probability distribution is involved and each agent should play its recommended strategy regardless to what its opponents do. The system is immune to counter-speculation.

Definition 4. *For a given normal form game* $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$ *and a strategy profile* $\mathbf{s}^*(\mathbf{t}) \in \times_{i=1}^n S_i$ *for a given type profile* $\mathbf{t}$, $\mathbf{s}^*(\mathbf{t})$ *is a dominant strategy equilibrium iff* $\forall i \in N, \forall \mathbf{s} \in \times_{i=1}^n S_i, u_i(t_i, \mathbf{s}) \leq u_i(t_i, \mathbf{s}^*)$.

To handle games of incomplete information into the graphical game representation, we associate each node of the directed graph with an utility function $u_i : \times_{i=1}^n S_i \mapsto \mathbb{R}$ in lieu of the local matrix M_i and a belief function p_i that determines the conditional probability $p_i(t_{-i}|t_i)$ using Bayes' rule. The motivation being that in many interesting cases, e.g., auctions, the utility matrix is obtained using a mathematical function that can be viewed as the composition of two mappings: the mapping from the type and strategy spaces of the players to the outcomes of the game and the mapping from the outcomes to real numbers. The utility function u_i can be simply encoded in an algorithmic way as a computer program and computes the utility $u_i(t_i, \mathbf{s})$ of player i when her neighbours play the strategies $\mathbf{s}$ in polynomial time. This representation captures normal form for Bayesian games by choosing G to be the complete graph and deriving the u_i and p_i from the original tabular form matrix. Furthermore, it allows us to succinctly represent games modelling auctions or markets where the utility and belief of each player are given by mathematical formulae.

For example, in a Vickrey auction, n agents have n valuations $v_i, i = 1, \ldots, n$ for a single item and m bids $b_i, i = 1, \ldots, m$. Denoting $\mathbf{b}$ the vector composed of $b_1, b_2, \ldots, b_m$, the utility $u_i(v_i, \mathbf{b})$ of agent i is given by the formula

$$u_i(v_i, \mathbf{b}) = \begin{cases} v_i - p_i & \text{if } b_i = \max(\mathbf{b}) \text{ and } p_i = \max(\mathbf{b}_{-i}) \\ 0 & \text{otherwise} \end{cases}$$

The graphical model of this game is a graph where each node i is connected to the $n-1$ remaining nodes and is associated with a simple function calculating the utility of agent i. We see that, in this case, the model does not exhibit a sparse structure and the graph is full but it avoids using matrices that can be large. For instance, if the bids are between 1 and 100, then we need to run the function for all possible inputs to build up the utility matrix of the tabular form, which contains 100^n entries. Because, there is an exponential number of possible input data, building up the utility matrices of the players is costly and may lead to storage problems. So the proposed refinement is useful for this kind of situation since it generalises the graphical model representation of games.

Definition 5. *A General Graphical Game is a data structure* $\{G, U, P\}$ *formed by*

- *an undirected graph* $G = (N, E)$ *wherein each node* $i \in N$ *represents a player and an edge* $(i, j) \in E$ *means player* i *influences the utility of player* j.
- *each node* $i, 1 \leq i \leq n$, *is associated to a local function* $u_i(t_i, \mathbf{s})$ *computing the utility of player* i *given its type and the strategies of its neighbours* $N_G(i)$ *and possibly to agent* i*'s belief distribution* p_i *over the set* $N_G(i)$. *The* u_i *and* p_i *form the sets* U *and* P *respectively.*

This general graphical game is also discussed in [22] but it is called General Normal Form game. In the rest of the paper, we assume for all $i \in N$, $|T_i| = |S_i| = d$ meaning d is the number of alternative strategies for each agent.

3 Verification of Equilibria for Normal Form Games

It is important to separate the verification of a given equilibrium point from its computation, which relies on an existence proof. The verification amounts to checking the properties that makes the point an equilibrium. Depending on the type of equilibrium, e.g., Nash, Bayesian or dominant strategy, the checking procedure may be computationally easy or hard.

Proposition 1. *Given a normal form game* $\mathcal{N} = (N, (S_i, u_i)_{i \in N})$ *and a strategy profile* $\mathbf{s}^*$*, verifying that* $\mathbf{s}^*$ *is a Nash equilibrium is in P.*

Proof sketch: Each player must solve a maximisation problem in which the feasible space of solutions is composed of d alternatives. For each agent, we can scan each alternative decision to find out its resulting utility given the optimal strategies of the other agents. It follows that the checking is in $\mathcal{O}(nd)$. □

Because computing a Bayesian Nash equilibrium is NP-complete for normal form games [22], we have the following result:

Proposition 2. *Given a normal form game* $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$ *and a strategy profile* $\mathbf{s}^*$*, verifying that* $\mathbf{s}^*$ *is a Bayesian Nash equilibrium is in P.*

Proof sketch: Given that in normal form, the game is represented by an exponential input data $q \equiv \mathcal{O}(d^n)$ with $d = |T_i|$. For each type t_i of player i, corresponds a strategy $s_i(t_i) \in S_i$. Checking equation (1) is linear in q for the player i. The checking for n players is therefore in $\mathcal{O}(nq)$. □

Proposition 3. *Given a normal form game* $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$ *and a strategy profile* $\mathbf{s}^*$*, verifying* $\mathbf{s}^*$ *is a dominant strategy equilibrium is in* $\mathcal{O}(nd^n)$*.*

Proof sketch: In the worst case scenario, the graph is full meaning each node is connected to the $n-1$ remaining agents. Scanning all different strategy profiles in order to check strategyproofness leads to a time complexity $\mathcal{O}(nd^n)$, which is exponential in the number of players. □

The question is to find out whether or not this checking can be carried out cheaply (in polynomial time). However, to verify a dominant strategy equilibrium, we can negate the property expressed by Definition 4, and then try to find a counter-example. In other words, we are asked to find out if

$$\exists i \in N, \exists \mathbf{s} \in \times_{i=1}^{n} S_i, u_i(t_i, \mathbf{s}) > u_i(t_i, \mathbf{s}^*) \tag{2}$$

The recommended strategy profile $\mathbf{s}^*$ is a dominant strategy equilibrium if and only if equation 2 cannot be satisfied. As a result, the problem of verifying a dominant strategy equilibrium for a normal form game can therefore be formulated as follows.

Verifying Dominant Strategy Equilibrium (VDSE)
Instance: A normal form game $\mathcal{N} = (N, (T_i, p_i, S_i, u_i)_{i \in N})$ and a recommended strategy profile $\mathbf{s}^*$ for a type profile $\mathbf{t}$.

Question: Does there exist a strategy profile $\mathbf{s}$ for which there exist an agent's utility $u_i(t_i, \mathbf{s})$, which is greater than $u_i(t_i, \mathbf{s}^*)$.

Theorem 1. *VDSE is NP-complete.*

Proof: Assume each agent i's utility function u_i is an integer linear function meaning $u_i(\mathbf{t}, \mathbf{s}) = \sum_{j=1,n} z_{i,j} t_j + \sum_{j=1,n} w_{i,j} s_j$ where $z_{i,j}$ and $w_{i,j}$ are integer values. Denoting $A = [Z, W]$ the $n \times (n+n)$-matrix resulting from the horizontal concatenation of the matrices $Z = (z_{i,j})_{1 \leq i,j \leq n}$ and $W = (w_{i,j})_{1 \leq i,j \leq n}$ and $\mathbf{x} = [\mathbf{t}, \mathbf{s}]$ the $2n$-vector resulting from the concatenation of the two n-vectors $\mathbf{t}$ and $\mathbf{s}$, the utility vector function $\mathbf{u} : (\times_{i=1}^{n} T_i) \times (\times_{i=1}^{n} S_i) \rightarrow \mathbb{Z}^n$ that computes the utilities for the n agents given their types and strategies can be written as the integer linear equation $\mathbf{u}(\mathbf{x}) = A\mathbf{x}$. Given the types $\mathbf{t}$ and the recommended strategy profile $\mathbf{s}^*(\mathbf{t})$, we can compute $\mathbf{u}^* = A\mathbf{x}^*$, where $\mathbf{x}^* = [\mathbf{t}, \mathbf{s}^*]$. Then, the problem is reduced to finding a vector $\mathbf{x}$ that satisfies $A\mathbf{x} > \mathbf{u}^*$. This gives rise to the following integer programming problem:

Integer Programming Instance: An $n \times m$-matrix A of integers and an integer n-vector $\mathbf{b}$.
Question: Does there exist a m-vector $\mathbf{x}$ of integers such that $A\mathbf{x} \geq \mathbf{b}$? This problem is NP-complete [23, p. 245]. This result holds even when the sets S_i of strategies are equal to $\{0, 1\}$ in which case we have the $0-1$ integer programming problem. □

Note that if the matrix Z is the $n \times n$ identity matrix, we recover the utilities u_i of the form $u_i(t_i, \mathbf{s})$. Moreover, compared to the problem of computing a Nash equilibrium wherein its existence is always guaranteed, we may not find a vector $\mathbf{x}$ that satisfies the inequality $A\mathbf{x} > \mathbf{u}^*$. Therefore, VDSE cannot be PPAD-complete. A class of normal form games in which the entries of the utility matrices are $0, 1$ are called *win-lose* games.

Corollary 1. *Verifying a dominant strategy for a win-lose game is NP-complete.*

However, if the graphical game representation is sparse in such a way that makes economic to exploit its structure, we have the following proposition.

Proposition 4. *Given a graphical game $\{G{=}(N, E), M\}$, if there exists a constant $k << n$ such that $|N_G(i)| \leq k$ for all node $i \in N$, then VDSE is in P.*

Proof: If there exists k such that $|N_G(i)| \leq k$, then the graph G is composed of subgraphs G_j containing at most k nodes. The players in G_j influence each other's utility meaning each $u_i(s_1, \ldots, s_n)$ of a player $i \in G_j$ can be rewritten as a function of only the set of strategies of the k players in G_j, say, $u_i(s_1, \ldots, s_k)$. The dominant strategy property can be checked independently for each player in the subgraph G_j in $\mathcal{O}(d^k)$. It follows that VDSE's complexity for this case is in $\mathcal{O}(nd^k)$. Note that this makes more sense in terms of computational savings when k is very small compared to n. □

4 Complexity Issues for General Games

So far, we have considered normal form games as basis for the verification of equilibria. Given a game mechanism, any verification of its properties required to be decidable. For that purpose, we restrict our study to finite games represented using the General Graphical Game as extended in Section 2.2. Typically, the game is represented as a directed graph in which a node i represents agent i and stores its utility function u_i and optionally its belief distribution function p_i. An edge (i, j) means agent i's strategies influence the utility of agent j. To focus on the verification procedure, we first assume that the utility and belief functions are computable.

In [24], it is pointed out that properties such as incentive compatibility or strategyproofness cannot be guaranteed when the equilibrium point is approximated, meaning it is in the vicinity of the optimum within a radius $\epsilon > 0$. The issues of approximately efficient winner determination algorithm and approximate strategyproofness are discussed for example in [25] in the case of a combinatorial multi-unit auction. To ensure incentive compatibility or strategyproofness, the game mechanism must have a winner determination algorithm that is exact. The utility of each player needs be accurate. We can ensure this required accuracy by considering:

- game mechanisms that are tractable since we avoid using approximate algorithms to compute their equilibrium points,
- small scale games that terminate within a reasonable amount of time despite their winner determination algorithms being exponential,

and by implementing those algorithms in exact arithmetic. To focus on the complexity of the verification procedure, we assume the winner determination algorithm is deterministic (no randomisation is allowed) and tractable. Consequently, the utility u_i of agent i can be computed with a polytime cost, which we denote $q \equiv \text{Cost}(u_i)$ for all functions u_i. Furthermore, when it is provided, we assume the cost of computing the probability $p_i(t_{-i}|t_i)$ is constant (a reasonable assumption). The question is how complex is it to verify an equilibrium for a finite game encoded in the general graphical game representation?

Note that we can build the normal form of the game compactly represented by the general graphical model by running the utility functions u_i over all possible outcomes. However, if there are two alternative strategies and n agents, then there are 2^n possible inputs for the functions u_i. Therefore, building up the normal form game is exponential in the number of agents. We directly analyse the verification procedure by using the General Graphical Game as a compact representation of the game. We derive the following simple results.

Proposition 5. *Given a finite game represented as a General Graphical Game* $\{G, U\}$ *and a strategy profile* $\mathbf{s}^*$*, verifying* $\mathbf{s}^*$ *is a Nash equilibrium takes a time complexity of* $\mathcal{O}(ndq)$.

Proposition 6. *Given a Bayesian game represented as a General Graphical Game* $\{G, U, P\}$ *and a strategy profile* $\mathbf{s}^*$*, verifying* $\mathbf{s}^*$ *is a Bayesian Nash equilibrium takes a time complexity of* $\mathcal{O}(nqd^n)$.

Proof sketch. The exponential complexity comes from the fact that given a type t_i of agent i, the number of configurations for t_{-i} in equation (1) is d^{n-1}. □

This result is discussed in [9] wherein the verification algorithm is described and its time complexity is analysed. Note that this verification may not be polynomial since computing a Bayesian Nash equilibrium is complete for the class PP (aka majority-P) of languages that can be decided in polynomial time by a non deterministic Turing machine wherein the acceptance condition is that a majority of computation paths accept [22]. The class PP introduced in [8] is a subclass of PSPACE but contains the class NP. We are currently working on the complexity class of this verification problem. As seen in the case of the normal form games, using a similar argument in the general graphical game representation, we have the following:

Theorem 2. *VDSE is NP-complete for general graphical games.*

5 Conclusions

In this paper, we have studied the computational complexity of verifying equilibrium properties in strategic games. We first considered normal form games and its graphical game representation and showed that whilst verifying a Nash or a Bayesian equilibrium is tractable, the verification of dominant strategy equilibrium is NP-complete.

We then moved to more general games that are finite and their general graphical game representations. We showed that whilst verifying a Nash equilibrium is tractable, the verification of a Bayesian Nash equilibrium is exponential in the number of players. Moreover, the problem of verifying dominant strategy equilibrium remains NP-complete in this setting. Future work includes that of finding out if checking a Bayesian Nash equilibrium for general games can be polynomial and if not which complexity class it belongs to.

Acknowledgement

We thank the UK EPSRC for funding this project under grant EP/D02949X/1.

References

1. Deguet, J., Demazeau, Y.: A complexity based feature to support emergence in MAS. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 616–619. Springer, Heidelberg (2005)
2. Papadimitriou, C.H.: Game theory, algorithms, and the internet. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 1–3. Springer, Heidelberg (2001), `http://www.cs.berkeley.edu/~christos/`
3. Nash, J.: Noncooperative games. Annals of Mathematics 54, 289–295 (1951)
4. Daskalakis, K., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. Electronic Colloquium on Computational Complexity (ECCC) 115 (2005)

5. Daskalakis, K., Papadimitriou, C.H.: Three-player games are hard. Electronic Colloquium on Computational Complexity (ECCC) 139 (2005)
6. Chen, X., Deng, X.: 3-NASH is PPAD-complete. Electronic Colloquium on Computational Complexity (ECCC) 134 (2005)
7. Chen, X., Deng, X.: Settling the complexity of two-player nash equilibrium. In: FOCS, pp. 261–272 (2006)
8. Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. 48, 498–532 (1994)
9. Guerin, F., Tadjouddine, E.M.: Realising common knowledge assumptions in agent auctions. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Hong Kong, China, pp. 579–586. ACM Press, New York (2006)
10. Vorobeychik, Y., Reeves, D.M., Wellman, M.P.: Automated mechanism design in infinite games of incomplete information: Framework and applications (2007), http://www.cscs.umich.edu/events/decentralization07/Infinite%20Games%20of%Incomplete%20Information.pdf
11. Tadjouddine, E.M., Guerin, F.: Verifying dominant strategy equilibria in combinatorial auctions. In: CSD Report No AUCS/TR0704, University of Aberdeen, Computing Sciences Department, Aberdeen, AB24 3UE, Scotland (2007), http://www.csd.abdn.ac.uk/~fguerin/DSE2007April.pdf
12. Van der Hoek, W.: Knowledge, rationality and action. In: AAMAS 2004. Proceedings of the AAMAS'04, New York, USA (2004)
13. Gibbons, R.: A Primer in Game Theory. Prentice-Hall, Englewood Cliffs (1992)
14. Pauly, M.: Programming and verifying subgame perfect mechanisms. Journal of Logic and Computation 15, 295–316 (2005)
15. Guerin, F.: An algorithmic approach to specifying and verifying subgame perfect equilibria. In: GTDT-2006. Proceedings of the Eighth Workshop on Game Theoretic and Decision Theoretic Agents, Hakodate, Japan. AAMAS 2006 (2006)
16. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems (Safety), vol. 2. Springer, Heidelberg (1995)
17. Tadjouddine, E.M., Guerin, F.: Verifying equilibria in games of complete and perfect information represented by presburger formulas. In: Proceedings of the 2006 LOFT Conference, Liverpool, UK, pp. 219–226 (2006)
18. Jackson, D.: Automating first-order relational logic. In: SIGSOFT FSE, pp. 130–139 (2000)
19. Jackson, D.: Alloy: A lightweight object modelling notation. ACM Trans. Softw. Eng. Methodol. 11, 256–290 (2002), http://alloy.mit.edu
20. Parkes, D.C.: Auction design with costly preference elicitation. Annals of Mathematics and AI 44, 269–302 (2004)
21. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical models for game theory. In: UAI '01. Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, pp. 253–260. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
22. Gottlob, G., Greco, G., Mancini, T.: Complexity of pure equilibria in bayesian games. In: Veloso, M.M. (ed.) IJCAI, pp. 1294–1299 (2007)
23. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman And Company, New York (1979)
24. Nisan, N., Ronen, A.: Computationally feasible VCG mechanisms. In: ACM Conference on Electronic Commerce, pp. 242–252. ACM Press, New York (2000)
25. Kothari, A., Parkes, D.C., Suri, S.: Approximately-strategyproof and tractable multi-unit auctions. Decision Support Systems 39, 105–121 (2005) (Special issue dedicated to the Fourth ACM Conference on Electronic Commerce)

Component-Based Development of Secure Mobile Agents Applications

Alvaro Moratalla and Sergi Robles

Department of Information and Communications Engineering,
Universitat Autònoma de Barcelona,
08193 Bellaterra - Spain
{amoratalla,srobles}@deic.uab.cat

Abstract. In this paper, we present a scheme for the design of mobile agents applications based on components. There is still a paramount issue to overcome in mobile agent technology to allow its popularization: the high programming complexity. The proposed scheme enables secure mobile agent creation by the composition of code components, significantly simplifying the work for the developer. Agent features like itinerary protection, results retrieving, or fault-tolerance can be achieved through reusable components. Our proposal is a steady step forward for the promotion of mobile agent applications.

Keywords: Mobile Agents, Agent development, Agent Architectures, Component-based development.

1 Introduction

One of the main obstacles hindering mobile agent technology to step forward is that the aided development of secure mobile agent applications is still on its early stages. Even though there has been several breakthroughs in the area of security on the last years [8], partially solving one of the most demanded problems, little or no effort at all has been done on working out a solution for the problem of secure mobile agent applications building.

When the application developer is programming mobile agents, she sadly discovers that a multitude of tasks not directly related with her application must be programmed as well. Besides of coding the main thread of the application, also protection mechanisms, outcome retrieval, tracking, or other features are to be included within the agent. This is a two folded problem: firstly, the developer faces up to spend more time programming all these side-features than the main code; secondly, the developer needs to be an expert in many areas to build her final mobile agents. One of the root causes for this, is that a large part of the agent-supporting code is not in the execution environment but need to be in the very agent: agent-driven mechanisms add flexibility and interoperability to mobile agent applications [8]. There is a need then to allow different experts to independently program all these off-the-shelf varied natured tasks, and let the developer include them on her program as required. This would significantly simplify the development of applications based on mobile agent technology.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 113–122, 2007.

There is a variety of development platforms that try to simplify agent's construction. The problem is that for the most part, they are limited to simplify the creation of the code of the application itself. In [6], for example, the tool presented allows the creation of mobile agents code using patterns. In [5] and in [7] high-level languages are presented that enable the application programming. In all these cases only the programming process of the application itself is simplified. All the side-features of agents are not taken into consideration. [9] presents a graphical development environment that simplifies the construction of agents in a graphical fashion. It is true that in most of the reviewed proposals the programming of the application as a whole is simplified, but they all consign to oblivion the complexity of mobile agent development. The natural modularity of mobile agents must be taken into account, realizing that most of the final code is side-functionality that can be developed independently such as fault-tolerance or result retrieval.

In [1] is presented a way to structure mobile agents in two basic blocks. In one hand, the agent *itinerary*, formed by the route of the agent and the code to be executed in each node of the route. This code can be protected by security mechanisms. In the other hand, the *control code*, that is, the code that manage the *itinerary* structure. The aim of this code is to obtain the code to be executed in each node of the agent route, decoding or decrypting it if needed. The code stored inside the *itinerary* is the code of the application, whilst the *control code* only process it for execution. This *control code* could be considered not part of the application, for it does not contribute to its functionality. This is the basis of the self-protected mobile agents.

This way of structuring mobile agents allows to automate their development. In [2] was presented a mobile agent builder that was able to build an executable protected mobile agent from specifications. The main idea was to separate the development of the application code from its protection. Using specifications of cryptographic architectures, they were able to build, in one hand, the *itinerary* structure, conveniently protected; and in the other hand, the *control code* specific for this cryptographic protocol.

This division of an agent changes the classic view of a mobile agent as a monolithic block of code. Nevertheless, the addition of another features as fault-tolerance or results recovery continues concerning to the agent programmer. The aim of this paper is to present a new way to develop mobile agents, extending the security solution drafted in [2] to the rest of agent features, and produce a real environment for the integral development of mobile agent secure applications. Our research wants to release the mobile agents developer from the programming of all these added features, that could be automatically picked from a repository and automatically integrated.

The automation of the mobile agent developing gives two great improvements to mobile agent technology. First, it frees the mobile agent developer from a large extent of burden work, allowing her to focus her work into the development of the agent behavior. That lead us to the second improvement: now, the code that implements all these additional features is independent and can be part of a global repository of well tested code. The developer can choose from any of these third-party off-the-shelf components and embed them into her applications. Tasks like agent's outcome protection and retrieval are now at the reach of any agent developer.

2 Building Mobile Agents by Components

Our scheme, SMARD (Secure Mobile Agent Rapid Development), is a development environment intended to simplify the programming of applications based on secure mobile agents. SMARD is comprised by three main tools: an Agent Designing Tool, an Agent Builder and an Agent Launcher.

The Agent Designing Tool (ADT) provides a graphical interface to aid the programmer in the definition of mobile agent itineraries. This graphical interface is divided in tabs that allow the programmer to define itinerary nodes, implement their tasks and see the messages generated by the agent compilation. Moreover, the ADT allows the mobile agent programmer to select which side-features she wants to add to her application functionality.

Once all nodes of the route and their corresponding tasks have been introduced in the Agent Designing Tool, this application generates an XML agent specification that includes an itinerary specification and a side-features selected specification. This XML document can be used to build a secure mobile agent that implements the side-features selected using the Agent Builder.

The mobile agents created by the Agent Builder are comprised by an explicit itinerary and a control code. On the one hand, the explicit itinerary is protected using cryptography, preventing potentially malicious platforms from accessing parts of the itinerary assigned to other platforms. On the other hand, the control code is built from libraries so that the agent itself can handle the security mechanisms applied to the itinerary and the side-features selected.

Once the agent is generated, it can be launched to the first platform of its itinerary using the Agent Launcher. For this purpose, the Agent Launcher exchanges a set of ACL messages with the remote immigration module, thus transferring a serialized instance of the agent and its associated code. If this migration protocol is successful, the agent execution starts on the remote platform.

2.1 Components Based Approach

In this scheme, the agent still is formed by the same two blocks. However the control code block is not a one-piece code any more, but it is formed by a set of independent code components. The former control code is now only one of the components of the current control code, specifically, the component commissioned to extract the local code from the itinerary and launch it.

In our scheme, the agent programmer can select in the ADT the features that she wants to add to the agent: Tracking, results retrieval, fault-tolerance, *etcetera*. Then, the agent builder will add to the agent all the code that implement these features. Finally, the agent builder will create a main component that will call this code to be executed. This main component code is the unification element of the control code. The result is a set of code components that will be executed sequentially. This structure is shown in figure 1.

This scheme is highly extensible. New components can be added without modifying any of the tools. The new components will be shown to the agent developer inside the ADT in order to allow her to select them to be a part of her agent.

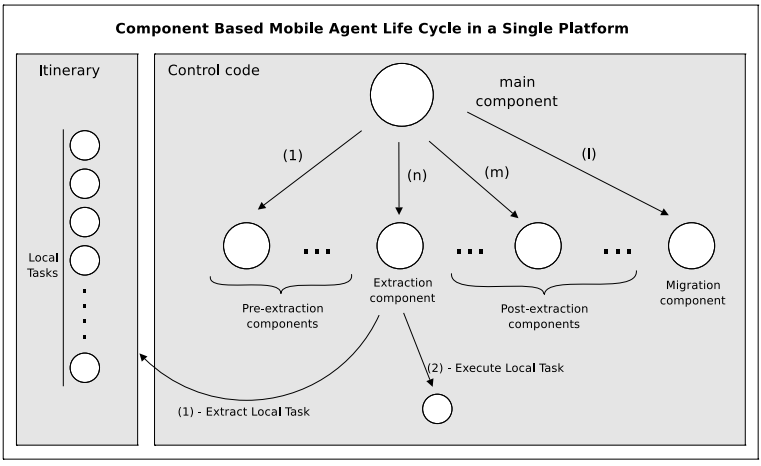


Fig. 1. Agent structure

For one single feature there can exist several implementations. For example, different fault-tolerance policies. The agent builder will add the code of one or another implementation in function of the agent programmer selection. Different experts can program all this code. We can see this scheme in figure 2. It is important to realize that different components perform different tasks, so they are fully independent.

2.2 The Extraction Component

The extraction component is a special component. We follow the approach drafted in [2], the itinerary (route and tasks) of the agent is stored in a particular structure. There are many ways to structure an itinerary. These structures may include protection mechanisms, but this is not necessary in all the applications. At the moment of the creation of an agent one structure must be selected, and the itinerary of this particular agent will be built following the rules of this particular structure. Then, when the agent arrives to a new platform, it has to execute a extraction component that understands the way the itinerary is structured in order to extract the task that must be executed in the current platform. This task might require cryptography if the structure that was selected used some protection. The code interpreting the structure and extracting the task to be executed is totally dependant of the code built the structure. Because of this, both codes should be programed at the same time.

2.3 The Components Order

As we have the control code split into a set of code blocks, there must be an order to execute then. For example, the code that executes the migration to the next platform will be logically the last code to be executed. That is because the migration begins when all the other tasks of the agent in the current platform have finished. A way to define this order is therefore required.

We can assign an ordinal number to each component to define its order. But this solution is weak because it is an obstacle for the extensibility. The addition of a new component will requires the redefinition of all the others components ordinals. For example, we have five components with an assigned ordinal number from 1 to 5, and we want to add a new component. To insert the component in the optimal order, we have to have some kind of knowledge about the others components to do it right, and these components might have been programmed by different persons. This collisions the extensibility of the scheme and the components independence. If the order is not the correct, there can be a malfunction in the agent execution. For example, if the migration component is executed before any other component, this last component will not be executed anyway.

The solution we propose is to define a set of stages to execute the components in relation to the agent's execution time in each agency. We can see these execution stages in figure 1. The extraction component is directly assigned to the local task execution, so its order is always fixed and does not require an explicit order specification. The migration component order is always fixed because is the last component to be executed. The order for executing the rest of components will be determined by these two.

In the components library, each component will be stored with a assigned order label that will allow the agent builder to create the main component in function of this order. These labels are:

pre-extraction. The component must be executed before the extraction component extracts the local task code and launches it.

post-extraction. The component must be executed after the extraction and execution of the local task code.

parallel. The component will be launched in a parallel thread at the rest of the agent execution. An example of this case may be an agent location code.

This definition of the components execution stages is enough general to allow the correct order location of the components. If there is more than one component in the same stage, the order they are executed is not relevant.

2.4 The Development Scheme Roles

The main improvement of the scheme we present is to separate the tasks of programming the agent itinerary (route and tasks), from the components programming.

In our development scheme, there are three basic programmer roles:

The agent programmer. The person in charge of programming the tasks that the agent must execute in each node of its route in order to achieve its objectives.

Component programmer. They will create the components that the agent programmer will select to load in her agent. They will program the supporting components too if required.

Security expert. They will create the code that will store the agent itinerary inside a structure, that may be protected, and the component that will extract the tasks to be executed in each node from this .

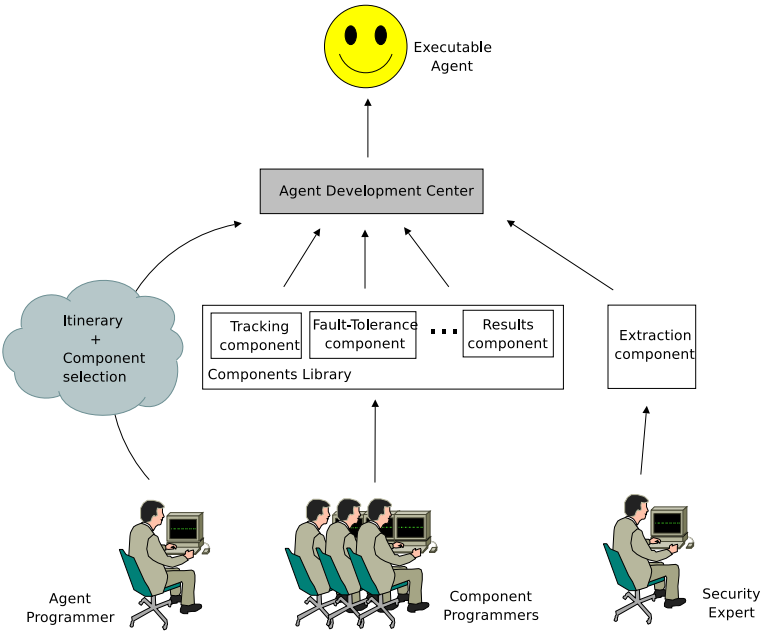


Fig. 2. The programming roles

The component programmers will build the code that implements different features. Each programmer can be an expert in each topic. When a new feature is required, the programmer will create the code that implements that feature. If the programmer respects the specifications her component will be fully compatible with our scheme. Then, this component will be stored in the components library, with a assigned order definition.

Once a new component is available, the agent programmer can select it to form part of the agent she is developing. She will use the Agent Designing Tool (ADT), that as well as allow the creation of the agent itinerary, also allows to select which components she wants to add to her agent. So that, the way these components become part of the agent is transparent to her, the agent builder will build the final agent.

2.5 The Components Library

In our scheme, the components must be available by the Agent Builder(AB), and the Agent Designing Tool (ADT). With the information stored, the ADT can show the agent programmer all the available components, so she can select the components that he wants to add to the agent.

The ADT builds an *XML* specification document with the itinerary and the selected components specification. With this information, the Agent Builder will build the control code including all the components. Firstly, it inserts in the agent the code of the components and then builds a main code that will call these code in the appropriate order. Moreover it has to build the itinerary and the structure component as described in [2].

There are agent components that must require a local counterpart in order to achieve its objectives. For example, a particular results retrieval component may require to communicate with an application located in a specific computer. This application is linked with this particular component implementation, so it must be implemented at the same time that the agent component. In fact there are some components that consist of two parts. This code will be stored at the same library and will be executed by the Agent Support Center. We will call these components *supporting components*. The details of this scheme are developed in the next section.

3 The Agent Support Center

There are components that may require an interaction with the agent owner. For example, the results retrieval component. In our scheme, there is an external facility to aid the process of agent management. This is a supporting center that allows the launched agent supporting by its owner. This center runs agents that interact with the launched agents, using ACL messages and IEEE-FIPA [3] protocols. These agents are called the *supporting agents*.

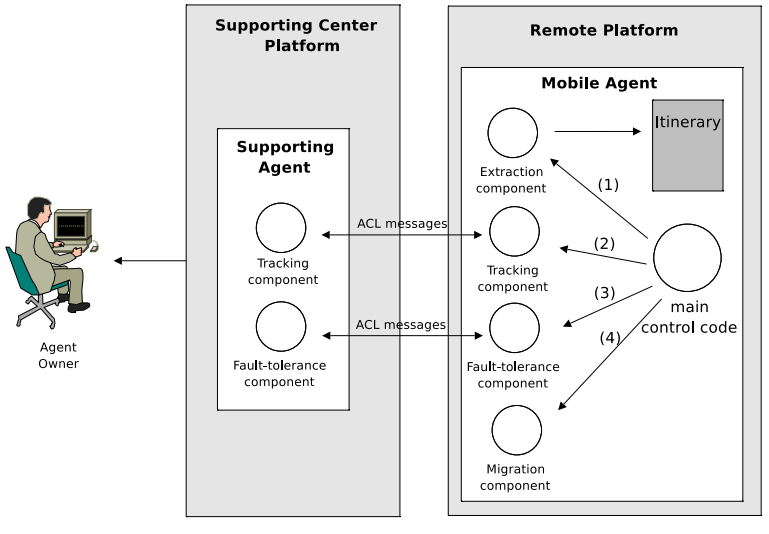


Fig. 3. Support center

There must be a *supporting agent* for each agent launched. All the supporting agents run in a local platform. A supporting agent is formed by the supporting components of each component loaded in the launched agent. These supporting components are the counterpart of the of the protocols implemented by the managed agent components. These components can be used, for example, to track the location of the agent during its execution.

The functionality presented by the supporting agents is directly linked with the components that can be selected to be part of the agent code. Not all the components require a supporting component that manages their features. A concrete fault-tolerance component, for example, must not requires any communication with a local supporting component, so in this case there is not counterpart in the supporting agent. If there is not any component that requires a supporting counterpart, any supporting agent will be built. We can see a scheme of that in figure 3.

In fact, the functionality is not provided by the supporting agent, but by the components executed. The natural way to communicate with an agent is using IEEE-FIPA protocols based on ACL messages. So, in our system, local agents are used to implement these protocols. The supporting center will be an agent platform in the computer of the agent owner.

At the moment of launching an agent to a remote platform to start its execution, a supporting agent with the supporting components loaded must be launched automatically in the local platform. These components can now receive and send ACL messages to its supported agent. The concrete protocols of this communication are completely open to the components programmers, who will program both parts of the protocol.

4 Implementation

The scheme presented on this paper has been implemented in [10].

In this implementation, SMARD agents run on JADE [4] platforms that have the Inter-Platform Mobility Service installed. At the moment, there is an extraction component, a migration component, a tracking component and two result retrieval components, one of them send the results to the support center, and the other sends the results to an e-mail account. In our implementation, the components are implemented as JADE behaviours and they are stored in a component library. Each component is defined by a file that stores the next information:

Component principal class. This class has to be loaded in the agent JAR, an instance of this class must be called from the main-component in order to start the component.

Order definition. The execution order of the component (pre-extraction, post-extraction, parallel).

Description. A description of the functionality of the component, this information is shown in the IDT to the agent developer at the time of selecting which components she wants to load in the agent.

Component auxiliary classes. A set of auxiliary classes to be loaded in the agent JAR. If the component is formed by more than one class. These classes are called from the principal class, so they don't need any order specification.

Supporting component class. The class of the control component to be loaded in the control agent. If the component has a supporting counterpart.

Supporting component auxiliary classes. A set of auxiliary classes of the control component. If the supporting component is formed by more than one class.

With this information, the ADT shows to the developer a set of components that she can select to be a part of her mobile agent application. The developer defines a route

for her mobile agent application and assign the code to be executed in each platform, besides a protection mechanism must be selected. The ADT generates an specification that is used by the Agent Builder (AB) to create an executable agent, that will be protected complying the protection mechanism selected and that will carry the code of the components selected by the developer. If any of these components requires a supporting counterpart, that will be loaded in a supporting agent.

Finally, the Agent Launcher (AL) will send the agent to the first platform of its route using the immigration module of the Inter-Platform Mobility Service. For this purpose, the Agent Launcher exchanges a set of ACL messages with the remote immigration module, thus transferring a serialized instance of the agent and its associated classes stored in a JAR file.

Once the agent is launched, the agent will travel through its itinerary performing the application it is programmed for. During its execution, it will exchange information with its supporting agent such as tracking information or results.

5 Conclusions

One of the most difficult challenges a secure mobile agent application developer must face at present is the programming of complex agents. She has to program entirely the agents, including protection mechanism, tracking control, or other advanced features, as well as basic methods such as outcome retrieval.

In this paper we have introduced a new development scheme for secure mobile agent applications based on components. By using it, the application developer can build her agents from a group of off-the-shelf re-usable components. Thus, this scheme is a important contribution that facilitates the popularization of secure mobile agent applications.

The scheme presented is based in building the final agent by the composition of different code components which implement different side-features. The agent developer can select these components at her convenience from a repository, depending on the wanted agent functionality. These code components are added to the original agent code automatically in the development process, and seamlessly to the programmer. Our environment allows the creation of mobile agents in a user friendly way.

All components can be independently developed by third parties, thus allowing the creation of common repositories. the components code will travel with the agent, but some components must need some supporting code. This is the case, for example, of the results retrieval where not only it is necessary to send the outcome of the agent but also to receive it. By using our scheme, a supporting agent is launched that implements all the supporting components related to the components loaded in the agent.

This scheme speeds up the mobile agent applications development. A full-featured agent can be built in a short time period. Once all platform tasks are implemented, the developer only has to select the components that she wants to include into her agent to obtain the final application. The components within the final agent implement all the desired features for the mobile agent application.

The developing scheme based on components is highly extensible. If new components are built, they can be added to the repository and made available for the rest of the community.

We have implemented this development scheme, the SMARD environment that consists of a set of tools: the Agent Designing Tool, the Agent Builder, and the Agent launcher. This implementation has shown the feasibility of our proposal, allowing the rapid development of secure mobile agent applications. This implementation allows the addition of new components. This leads us to continue toward the completion of a full development environment, including several security related features such as fault tolerance or anonymity.

Acknowledgments

This work has been partially funded through the Spanish project TSI2006-03481, and the support of the Catalan project 2005-FI-00752.

References

1. Ametller, J., Robles, S., Ortega, J.A.: Self-Protected Mobile Agents. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 362–367. ACM Press, New York (2004)
2. Garrigues, C., Robles, S., Moratalla, A., Borrell, J.: Building Secure Mobile Agents using Cryptographic Architectures. In: 2nd European Workshop on Multi-Agent Systems, pp. 243–254 (2004)
3. IEEE FIPA: Foundation for intelligent physical agents (2006), `http://www.FIPA.org`
4. JADE: JADE, Java Agent DEvelopment Framework (2004), `http://jade.cselt.it`
5. Li, X., Cao, J., He, Y.: A language for description and verification of mobile agent algorithms. In: CIT'04. Proceedings of the The Fourth International Conference on Computer and Information Technology, IEEE Computer Society, Los Alamitos (2004)
6. Modak, V.D., Langan, D.D., Hain, T.F.: A pattern-based development tool for mobile agents. In: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, pp. 72–75. ACM Press, New York (2005)
7. Mueller, W., Meyer, A., Zabel, H.: A Language for the Rapid Prototyping of Mobile Evolving Agents. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences HICSS'01, IEEE Computer Society Press, Los Alamitos (2001)
8. Robles, S., Ortega-Ruiz, J.A., Ametller-Esquerra, J.: Security Solutions and Killer Applications: Unlocking Mobile Agents. British Computer Society Expert Update 7(3), 18–23 (2004)
9. Sharma, R.: Mobile agent construction environment. Master's thesis, Thayer School of Engineering, Dartmouth College (June 1997)
10. SMARD: Secure Mobile Agent Rapid Development. SourceForge Project (2007), `http://sourceforge.net/projects/smard/`

Design Patterns for Self-organising Systems

Luca Gardelli, Mirko Viroli, and Andrea Omicini

Alma Mater Studiorum–Università di Bologna
Via Venezia 52 - 47023 Cesena, Italy
{luca.gardelli,mirko.viroli,andrea.omicini}@unibo.it

Abstract. Natural systems are regarded as rich sources of inspiration for engineering artificial systems, particularly when adopting the multiagent system (MAS) paradigm. To promote a systematic reuse of mechanisms featured in self-organising systems, we analyse a selection of design patterns devised from the self-organisation literature. Starting from our reference MAS metamodel, we propose a pattern scheme that reflects the peculiarities of self-organising systems. Then, we provide a complete characterisation of each pattern, with particular attention to the problem description, the solution with respect to our metamodel, the natural systems which have inspired the pattern and known applications.

1 Introduction

Self-organisation is a very compelling approach to the engineering of complex systems because of the many interesting properties, including adaptivity and robustness. Although, forward engineering of self-organising systems, i.e. finding the individual behaviour to meet the desired global dynamics, rapidly becomes unfeasible as the complexity of the system increases. Hence, it is becoming common practice to exploit existing models of natural systems, particularly social insects: these models provide a characterisation of global dynamics with respect to individual actions and environmental parameters. To ease this process, we promote the use and development – since few works exist about this topic [1,2] – of design patterns for self-organising systems to establish a mapping between artificial systems problems and natural systems solutions. First introduced in 1977 by Alexander in architecture [3], the concept of *design pattern* later gained wide consensus in computer science with the object-oriented paradigm [4]. A design pattern provides a reusable solution to a recurrent problem in a specific domain: it is worth noting that a pattern does not describe an actual design, rather it encodes an abstract model of the solution using specific entities of the paradigm in use. Multiagent system (MAS) researchers synthesised patterns for the agent paradigm, providing solutions related to resource access, mobility and basic social skills [5,6,7]. The use of design patterns offers several advantages, such as, reducing design-time by exploiting off-the-shelf solutions, and promoting collaboration by providing a shared language. Specifically, in the case of self-organising systems, patterns play a key role in driving the designer choices among the chaotic behaviours displayed by complex systems.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 123–132, 2007.

To promote the acceptance of new patterns, as well as to reduce ambiguity, it is necessary to frame a pattern with respect to a shared scheme, ensuring a good degree of coherence of the whole pattern catalogue. For example, patterns for the object-oriented paradigm are described according to the scheme provided in [4]: although, as pointed out in [8,7], since the agent paradigm cannot be effectively characterised using only object-oriented abstractions, patterns for MAS should be described using specific schemata. In particular, a candidate scheme should reflect the peculiarities of the target MAS metamodel: to this purpose, several pattern classification and schemata have been proposed [8,9]. However, pattern schemata for MAS, like the one in [8], do not adequately capture the peculiarities of self-organising systems, namely, which are the forces responsible for the feedback loop, and which notion of locality/topology is needed. Furthermore, to our knowledge, no specific pattern scheme has been proposed to deal with the previous aspects, and the few existing works rely on existing MAS schemata.

The contribution of this article is twofold: we extend current pattern schemata to better represent self-organising MAS, and characterise some patterns with respect to our MAS metamodel. The remainder of the article is structured as follows: Section 2 describes our MAS metamodel based on agents and environmental resources, i.e. artifacts. Section 3 describes the reference pattern scheme, then Section 4 analyses each pattern, namely, Collective Sort, Evaporation, Aggregation and Diffusion. Section 5 concludes by providing final remarks and listing future research directions.

2 Our Reference MAS Metamodel

When modelling natural systems and developing artificial ones, the MAS paradigm is the best choice since it provides the suitable abstractions for modelling and relating the entities. Although being recognised as unique, the MAS paradigm is captured from different perspectives in different metamodels emphasising specific features. We adopt the agents & artifacts metamodel (A&A), where a MAS is modelled in terms of two fundamental abstractions: *agents* and *artifacts* [10]. Agents are autonomous pro-active entities encapsulating control, driven by their internal goal/task. When developing a MAS, sometimes entities do not require neither autonomy nor pro-activity to be correctly characterised. Artifacts are passive, reactive entities providing services and functionalities to be exploited by agents through a *usage interface*. It is worth noting that artifacts typically realise those behaviours that cannot or do not require to be characterised as goal oriented [10]. Artifacts mediate agents interactions and support coordination in social activities and embody the portion of the environment that can be designed and controlled to support MAS activities [11,12].

Based on this metamodel, we recognise a recurrent solution when designing self-organising MAS: the solution depicted in Figure 1 is an architectural pattern. In a typical self-organisation scenario, agents perturb the environment, and while the environment evolves to compensate the perturbation, agents perception is affected, creating a feedback loop able to sustain the self-organisation

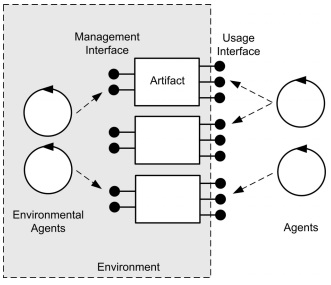


Fig. 1. Architectural pattern featuring environmental agents as artifact administrators

process. Hence, on one side we have agents exploiting artifacts services, that we name *user agents* from now on. Although, because of the enormous repertoire of behaviours that can be exhibited by user agents in different scenarios, we cannot further detail their role in the architecture. On the other side artifacts provide basic services in terms of simple local elaboration of agents requests. Since the passive/reactive attitude of artifacts, they cannot *autonomously* adapt their behaviour to meet the changing requirements of a dynamic and unpredictable environment. Hence, we envision MAS environments built upon artifacts and *environmental agents*: in particular environmental agents are in charge of those goal-oriented behaviour needed for the management of the artifacts. Specifically, we separate the usage interface from the *management interface* – in an Autonomic Computing style [13] – whose access is restricted only to environmental agents: furthermore environmental agents may exploit artifacts inspectability and malleability.

3 A Reference Pattern Scheme

The literature provides several pattern schemata, e.g. [4,9,8]. In a previous work [14], we analysed patterns for self-organising system relying on the scheme for MAS described in [8]: although, we recognise the failure to capture essential self-organisation aspects, namely forces involved in the feedback loop and topology notion. The pattern scheme we propose extends the one described in [8] and is summarised in Table 1, where the novel items are emphasised. Particularly relevant to this work are the *feedback loop* and *locality* elements:

Feedback loop. Describes the processes or actions involved in the establishment of a feedback loop, i.e. the actions providing positive and negative feedback. For example, in a digital pheromone infrastructure [15], the positive feedback consists in the agent depositing pheromones, while the environment provides the negative feedback in the form of pheromone evaporation.

Locality. Requirements in terms of spatial topology or action-perception ranges: if the environment has a notion of continuous space, perception range is specified as a float value; if the environment has a graph topology, ranges are specified as the number of hops.

Table 1. An extension to the pattern scheme described in [8]

Name	The name of the pattern
Aliases	Alternative names
Problem	The problem solved by the pattern
Forces	Trade-offs in system dynamics
Entities	Entities participating to the pattern
Dynamics	Entities interactions
Feedback Loop	Interactions responsible for the feedback loop
Locality	Describe the type of locality required
Dependencies	Environmental requirements
Example	An abstract example of usage
Implementation	Hints on implementation
Known Uses	Existing applications using the pattern
Consequences	Effects on the overall system design
See Also	References to other patterns

4 Patterns of Self-organising Systems

4.1 Collective Sort Pattern

Social insects tend to arrange items in their surroundings according to specific criteria, e.g. broods and larvae sorting in ant colonies [16,17]. This process of collectively grouping items is commonly observed in human societies as well, and serves different purposes, e.g. garbage collection. Also in artificial systems collective sort strategies may play an important role: for instance, grouping together related information helps to manage batch processing.

We consider our previous exploration of Collective Sort dynamics in a MAS context [18,19] in order to synthesise a pattern. From an arbitrary initial state, see Figure 2, the goal of Collective Sort is to group together similar information in the same node, while separating different kinds of information as shown in Figure 2b. Although, this is not always possible: indeed, if we consider a network having two nodes and three kinds of information, two of them are going to coexist on the same node. Due to random initial situation and asynchronous interactions the whole system can be modelled as stochastic. Hence, it is not generally known a priori where a specific cluster will appear: clusters location is an emergent properties of the system [18], which indeed supports robustness and unpredictable environmental conditions. Table 2 summarises the features of the collective sort pattern.

4.2 Evaporation Pattern

In social insects colonies coordination is often achieved by the use of chemical substances, usually in the form of pheromones: pheromones act as markers for

Table 2. A summary of the features of the collective sort pattern according to the reference scheme

Name	Collective Sort
Aliases	Brood Sorting, Collective Clustering
Problem	MAS environments that does not explicitly impose constraints on information repositories may suffer from the overhead of information discovery.
Forces	Optimal techniques requires more computation while reducing communication costs: on the other hand, heuristics allows for background computation but increase communication costs.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. The artifacts have to provide specific content inspection primitives depending on the implementation. Environmental agents monitor artifacts for new information, and depending on artifacts content may decide to move an information to a neighboring artifacts.
Feedback Loop	Positive feedback is determined by environmental agents moving items to the appropriate cluster, while negative feedback happens when an item is misplaced.
Locality	Either continuous and discrete topology are suitable. Larger perception range improve strategy efficiency, but perception of immediate neighborhood is sufficient, but requires memory of items encountered.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 2 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by an insertion action: either approaches are suitable and choice depend on performance requirements. Moving information requires an aggregated view upon artifacts content, e.g. using counters or spatial entropy measures: in the case this is not feasible or too expensive, content sampling techniques can be used, see [18] for a detailed discussion.
Known Uses	Explorations in robotics for sorting a physical environment [16].
Consequences	Collective Sort may not work when used in combination with other patterns that spread information across the MAS: in particular collective sort opposes to Diffusion (Section 4.4).
See Also	-

specific activities, e.g. food foraging [16,17]. Specifically, these substances are regulated by environmental processes called *aggregation*, *diffusion* in space and *evaporation* over time: each process can be captured by a specific pattern, hence, it is analysed separately. This class of mechanisms for indirect coordination mediated by the environment is called *stigmergy*, and it has been widely applied in the engineering of artificial systems [20,15,21].

Evaporation is a process observed in everyday life, although with different implications: e.g. from scent intensity it is possible to deduce amount and distance of its source. In the case of insect colonies, marker concentration tracks activities: e.g. absence of pheromone implies no activity or no discovered food source. In ant food foraging [17] when a food source is exhausted, the pheromone trail is no longer reinforced and slowly evaporates.

Evaporation has a counterpart in artificial systems that is related to information obsolescence [22,23]. Consider a web page listing several news: fresh information is inserted at the top of the page and news *fade* as time passes, which can be translated in visual terms in a movement towards the end of the page. In general, evaporation can be considered a mechanism to reduce information amount, based on a time relevance criterion. As an example, starting from an initial state – see Figure 3a – evaporation removes old information over time and, in absence of new information insertion, eventually erases everything—see Figure 3b.

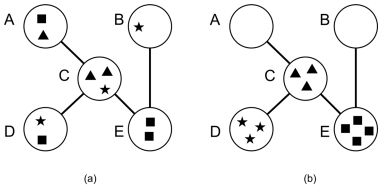


Fig. 2. Collective sort (a) an initial state (b) the final state

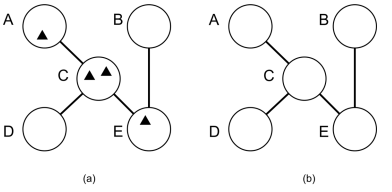


Fig. 3. Evaporation (a) an initial state (b) the final state with no reinforcement

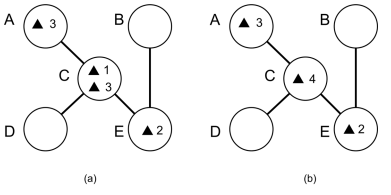


Fig. 4. Aggregation (a) an initial state (b) the final state

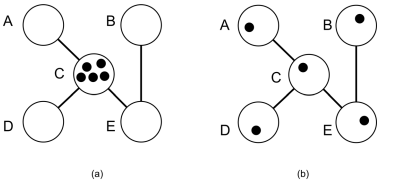


Fig. 5. Diffusion dynamics (a) an initial state (b) the desired final state

4.3 Aggregation Pattern

Pheromone deposited in the environment is spontaneously *aggregated*, i.e. separate quantities of pheromone are perceived as an individual quantity but with greater intensity [16,17], see Figure 4 for a visual example. Aggregation is a mechanism of reinforcement and is also observable in human social tasks [22]. The ranking mechanism is a typical example: when browsing the Internet someone finds an interesting fact, he/she can leave a (reinforcement) comment that is typically anonymously and automatically aggregated with comments of other users. It is then evident that, while evaporation is driven by the environment, aggregation is driven by the user agent. When used in combination with evaporation, aggregation lets the designer close a positive/negative feedback loop, allowing for auto-regulated system in self-organisation and Autonomic Computing [13] styles.

4.4 Diffusion Pattern

When pheromone is deposited into the environment it spontaneously tends to diffuse in neighboring locations [17]. This process, called *diffusion*, is omnipresent in nature and hence is studied in several fields under different names, e.g. osmosis in chemistry. Starting from an arbitrary state, see Figure 5a, diffusion eventually distribute the information equally across all nodes [1], see Figure 5b.

While aggregation and evaporation processes are often used in combination and act locally, diffusion can be used alone and requires a notion of topology. Furthermore, in diffusion the initial quantity of information is *conserved* but

Table 3. A summary of the features of the evaporation pattern according to the reference scheme

Name	Evaporation
Aliases	None to our knowledge.
Problem	MAS environments can soon become overwhelmed by information deployed by agents.
Forces	Higher evaporation rates release memory, but require more computation: furthermore, evaporated information cannot be recovered!
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. The artifacts assign a timestamp/counter to the received information. Environmental agents erase obsolete information/information whose counter reached zero: eventually, all the information is removed.
Feedback Loop	User agents deposit items in the environment while environmental agent evaporate them.
Locality	Perceptions and actions happens only locally. Either continuous and discrete topology are suitable.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 3 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements.
Known Uses	A fundamental element of stigmergy [17] and digital pheromone based application [20,15,21].
Consequences	-
See Also	When used in combination with Aggregation (Section 4.3) or Diffusion (Section 4.4), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 4. A summary of the features of the aggregation pattern according to the reference scheme

Name	Aggregation
Aliases	None to our knowledge.
Problem	Large scale MAS suffer from the amount of information deposited by agents, which have to be sifted in order to synthesise macro information.
Forces	Higher aggregation rates provide results closer to the actual environment status, but require more computation.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. Environmental agents look for new information and aggregate it with older information to produce a coherent result.
Feedback Loop	User agents deposit items in the environment while environmental agent synthesise an aggregated info.
Locality	Perceptions and actions happens only locally. Either continuous and discrete topology are suitable.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 4 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements. It is worth noting that aggregation is a very simple task and could be automatically handled by artifacts, when properly programmed: although, it is easier to have separate agents for different functionalities, which can be individually paused or stopped.
Known Uses	A fundamental element of stigmergy [17] and digital pheromone based application [20,15,21]. In e-commerce applications customers feedback is usually aggregated, e.g. average ranking, in order to guide other customers.
Consequences	-
See Also	When used in combination with Evaporation (Section 4.2) or Diffusion (Section 4.4), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

Table 5. A summary of the features of the diffusion pattern according to the reference scheme

Name	Diffusion
Aliases	Plain Diffusion, Osmosis.
Problem	In MAS where agents are only allowed to access local, agents reasoning suffer from the lack of knowledge about neighboring nodes.
Forces	Higher diffusion radius brings information further away from its source, providing a guidance also to distant agents: although, the infrastructure load increases, both in terms of computation and memory occupation. Furthermore, diffused information does not reflect the current status of the environment hence providing false hints.
Entities	The pattern involves artifacts, user agents and environmental agents.
Dynamics	User agents inject information in the artifacts. A weight is assigned to the information from artifacts or user agents. Environmental agents diffuse information decreasing the weights in local node and correspondingly increasing the weights in neighboring nodes.
Feedback Loop	User agents deposit items in the environment while environmental agent scatter them to neighboring locations.
Locality	User agents perceptions and actions happens only locally, while environmental agents need to perceive and act at least at one hop of distance. Either continuous and discrete topology are suitable.
Dependencies	It requires an environment compliant to the A&A metamodel.
Example	See Figure 5 for a visual example.
Implementation	Environmental agents may perform periodic inspection or been triggered by a specific event: either approaches are suitable and choice depend on performance requirements.
Known Uses	A fundamental element of stigmergy [17] and digital pheromone based application [20,15,21]. In e-commerce applications the *see-also* hint is a typical example of information diffusion were the topology is built upon a similarity criterion of products.
Consequences	Diffusion may not work when used in combination with other patterns that spread information across the MAS: in particular diffusion opposes to Collective Sort (Section 4.1).
See Also	When used in combination with Evaporation (Section 4.2) or Aggregation (Section 4.3), it allows for building complex behaviours: in particular, Evaporation + Aggregation + Diffusion is the Stigmergy pattern.

spatially spread: although, other forms of diffusion may be conceived to produce stable gradients [20].

5 Conclusion

In the engineering of systems with emergent properties, it is common practice to rely on existing models of natural activities. Despite the existence of many patterns for MAS – see [8,5,6,9] just to name a few – we currently lack a systematic patterns catalogue which could guide the designer of self-organising systems: few notable exceptions include [24,1,2]. In this article, we provide an extension to the pattern scheme described in [8] to better reflect the peculiarities of self-organising systems. Furthermore, we describe a few patterns devised from the self-organisation literature, namely, Collective Sort, Evaporation, Aggregation and Diffusion: each pattern has been analysed with respect to the proposed pattern scheme. Minor contributions of this article include

– recognising the need of a pattern catalogue and identifying the special role of patterns in the engineering of artificial self-organising systems;

– recognising the qualitative differences between patterns for self-organising systems with respect to MAS and object-oriented patterns;
– show how the composition of patterns is a challenging task: the evaluation of a specific pattern requires the knowledge about the interplay of the dynamics.

Future works include

– identifying and using the appropriate graphical notation for describing patterns;
– extending the pattern catalogue and validating the proposed pattern scheme.

References

1. Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G.A., Ducatelle, F., Gambardella, L.M., Ganguly, N., Roberto Montemanni, M.J., Montresor, A., Urnes, T.: Design patterns from biology for distributed computing. ACM Transactions on Autonomous and Adaptive Systems 1(1), 26–66 (2006)
2. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 28–49. Springer, Heidelberg (2007)
3. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Professional Computing. Addison-Wesley, Reading (1995)
5. Kendall, E.A., Krishna, P.V.M., Pathak, C.V., Suresh, C.B.: Patterns of intelligent and mobile agents. In: Sycara, K.P., Wooldridge, M. (eds.) AGENTS '98. 2nd International Conference on Autonomous Agents, pp. 92–99. ACM Press, New York (1998)
6. Aridor, Y., Lange, D.B.: Agent design patterns: elements of agent application design. In: Sycara, K.P., Wooldridge, M. (eds.) AGENTS '98. 2nd International Conference on Autonomous Agents, pp. 108–115. ACM Press, New York (1998)
7. Deugo, D., Weiss, M., Kendall, E.: Reusable Patterns for Agent Coordination. In: Coordination of Internet Agents: Models, Technologies, and Applications, pp. 347–368. Springer, Heidelberg (2001)
8. Lind, J.: Patterns in agent-oriented software engineering. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, pp. 47–58. Springer, Heidelberg (2003)
9. Cossentino, M., Sabatucci, L., Chella, A.: Patterns reuse in the PASSI methodology. In: Omicini, A., Petta, P., Pitt, J. (eds.) ESAW 2003. LNCS (LNAI), vol. 3071, pp. 294–310. Springer, Heidelberg (2004)
10. Ricci, A., Viroli, M., Omicini, A.: Programming MAS with artifacts. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Programming Multi-Agent Systems. LNCS (LNAI), vol. 3862, pp. 206–221. Springer, Heidelberg (2006)
11. Weyns, D., Omicini, A., Odell, J.: Environment as a first-class abstraction in multi-agent systems. Autonomous Agents and Multi-Agent Systems 14(1), 5–30 (2007)

12. Viroli, M., Holvoet, T., Ricci, A., Shelfthout, K., Zambonelli, F.: Infrastructures for the environment of multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1), 49–60 (2007)
13. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
14. Gardelli, L., Viroli, M., Omicini, A.: Design patterns for self-organizing multiagent systems. In: De Wolf, T., Saffre, F., Anthony, R. (eds.) 2nd International Workshop on Engineering Emergence in Decentralised Autonomic Systems (EEDAS 2007), ICAC 2007, University of Greenwich, London, UK, pp. 62–71. CMS Press, Jacksonville, FL, USA (2007)
15. Parunak, H.V.D., Brueckner, S.A., Sauter, J.: Digital pheromones for coordination of unmanned vehicles. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 246–263. Springer, Heidelberg (2005)
16. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems, Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York, US (1999)
17. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, USA (2001)
18. Casadei, M., Gardelli, L., Viroli, M.: Simulating emergent properties of coordination in Maude: the collective sorting case. Electronic Notes in Theoretical Computer Sciences 175(2), 59–80 (2007) (5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA 2006) (2006)
19. Gardelli, L., Viroli, M., Casadei, M., Omicini, A.: Designing self-organising MAS environments: the collective sort case. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) Environments for Multi-Agent Systems III. LNCS (LNAI), vol. 4389, pp. 254–271. Springer, Heidelberg (2007)
20. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the TOTA middleware. In: PerCom 2004. 2nd IEEE Annual Conference on Pervasive Computing and Communications, pp. 263–273. IEEE, Los Alamitos (2004)
21. Weyns, D., Schelfthout, K., Holvoet, T., Lefever, T.: Decentralized control of E'GV transportation systems. In: AAMAS 2005. 4th International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, July 25-29, 2005, pp. 67–74. ACM, New York (2005)
22. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive stigmergy: A framework based on agents and artifacts. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 124–140. Springer, Heidelberg (2007)
23. Parunak, H.V.D.: A survey of environments and mechanisms for human-human stigmergy. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 163–186. Springer, Heidelberg (2006)
24. Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case studies for self-organization in computer science. Journal of Systems Architecture 52(8–9), 443–460 (2006)

Experience with Feedback Control Mechanisms in Self-replicating Multi-Agent Systems

Sebnem Bora and Oguz Dikenelli

Computer Engineering Department
Ege University, Izmir, Turkey
sebnem.bora,oguz.dikenelli@ege.edu.tr

Abstract. In this paper, we present an approach for adaptive replication to support fault tolerance. This approach uses a feedback control theory methodology within an adaptive replication infrastructure to determine replication degrees of replica groups. We implemented this approach in a multi-agent system to survive Byzantine failures. At the end of the paper, we also provide some experimental results to show the effectiveness of our approach.

1 Introduction

Replication is a key technique to achieve fault tolerance in distributed and dynamic environments, by masking errors in the replicated component. A distributed application is a set of cooperating objects, where an object could be a virtual node, a process, a variable, an object, as in object-oriented programming, or an agent in multi-agent systems. When an object is replicated, the application has several identical copies of the object (called replicas). When a failure occurs on a replica, the failure is masked by its other replicas, therefore availability is ensured in spite of the failure.

As distributed systems, multi-agent systems (MAS) are vulnerable to failures resulting from the system crash or shortages of system resources, slow-downs or breakdowns of communication links, and errors in programming. Consequently, a fault in an agent may spread throughout a multi-agent system, and cause a degradation of the system performance and even the multi-agent system to fail. Replication is also applied as a key technique to achieve fault tolerance in multi-agent systems.

In replication-based approaches for achieving fault tolerance, there must be multiple copies of the same object (replicas). According to the preferred replication approach, all replicas of an object (a server or an agent) may run concurrently, possibly in different environments. The true cost of the replication of a single object is the sum of the cost of replica creation, replica usage, and overheads incurred by the coordination of the replicas. Moreover, a fixed number of resources in a system are reserved to provide redundancy for the performance of a certain task. However, usage of a fixed number of resources in a system can be expensive. Varying the replication degree, which is the number of replicas in a fault tolerant system, can reduce the cost caused by the replication. Adaptive fault tolerance enables a system to change its replication degree in accordance with its environment.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 133–142, 2007.

In this paper, we present an approach for adaptive replication. It can be applied to distributed systems in a general context, including multi-agent systems. This approach uses an observation mechanism and a feedback control mechanism within an adaptive replication infrastructure to support adaptive fault tolerance. The main strategy used in our approach is to insert control theory methodology and analysis into adaptive replication. We have implemented this approach as a part of fault tolerant multi-agent systems and evaluated it. In this paper, we present very interesting results. The results show that control theory methodologies can really improve fault tolerance in multi-agent systems.

Thus the paper makes the following contributions:

- It describes an adaptive replication mechanism for multi-agent systems that can survive Byzantine failures in asynchronous environments.
- It describes how we apply a control theory methodology in the adaptive replication mechanism.

The remainder of this paper is structured as follows: Section 2 presents the context of the work and introduces an abstract architecture for adaptive replication in MAS organizations; Section 3 briefly introduces SEAGENT overall architecture and describes how to support the development of fault tolerant multi-agent system using SEAGENT; Section 4 presents the feedback controller based adaptive replication infrastructure and how to realize adaptive replication; Section 5 gives the case study; and finally Section 6 gives the conclusion.

2 Context of This Work

In this work, agents are distributed over the network, and operate in open environments. Thus, we assume that the system is in an asynchronous environment and subject to message omissions; agent crashes (fail-silent), and network partitions. We also assume no message corruptions since the communication software is able to recognize them and reject faulty messages. In this work, we use a Byzantine failure model. If a failed system can behave arbitrarily, it is said to exhibit Byzantine failure behavior.

2.1 Types of Fault Tolerance Techniques Considered

There are two main approaches for replication: the passive replication approach (also called primary-backup) and the active replication approach. In active replication, there are extra copies of a service (called replicas) processing client requests and synchronizing internal states with every other peer service. If a primary service fails, any replica can become a primary service provider. In order to implement replica coordination in active replication, replicas must communicate with each other. Group communication provides multi-point-to-multi-point communication by organizing replicas in groups.

In the passive replication approach, there are extra copies (replicas) of a service. However, primary one responds to client requests. Primary periodically updates replicas' states. If primary fails, one replica can be elected as a primary service provider.

In our system, we have implemented both active and passive replication approaches. The system can use either the active replication approach or the passive replication approach. In case of using a Byzantine failure model, the system only uses the active replication approach. In the group, the algorithm of active replication works as follows:

1. A client that requests an action performed, sends a request to the primary.
2. The primary multicasts the request to its replicas.
3. The primary and its replicas perform the request. Replicas send the results to the primary.

In order to apply the algorithm, we assume that all replicas in a group are deterministic and they start in the same state.

Active and passive replication approaches mainly focus on coordination within the group. In addition to coordination requirements, the replication degree, which means the number of replicas within a group, is a critical concept for applying fault tolerance policies based on replication. The problem is how the system will decide on the number of replicas at runtime. The replication degree can be identified adaptively or statically. In a static fault tolerance policy, this number is set to the number defined by a programmer at initialization. In an adaptive fault tolerance policy, a replication manager decides on the number of replicas in a group based on resources of the system. This process incorporates an observation mechanism that transparently monitors the agents' behaviors as well as the availability of resources, and adaptively reconfigures the system resources.

In our approach, the replication degree of a group is identified adaptively. The replication manager decides on the number of replicas in a group based on resources of the system, criticality of the primary (leader) agent, and the agreement problem. To reach agreement in asynchronous environments, there must be 3f+1 replicas in a group when up to f replicas are faulty (find the proof in [1]). If we denote the number of replicas by R, then R must be equal to 3f+1. The primary waits for f+1 replies from different replicas with the same result. This is the result of the request.

2.3 An Abstract Architecture for Adaptive Replication

The proposed abstract architecture is illustrated in Fig. 1 and built on the FIPA (Foundations for Physical Agents) based MAS architecture [2]. As seen from the overall figure, this architecture uses aspects of both agent-centric and system-centric approaches. We move the services for fault tolerance into agents and the mechanism of monitoring into another software entity.

In order to achieve effective fault handling, we monitor the environment to collect data to estimate the most critical agents of the organization. The criticality of an agent is the measure of the potential impact of the failure of that agent on the failure of the organization. In order to collect the data, we associate an observation mechanism to another software entity called the "feedback controller based adaptive replication infrastructure". This infrastructure is implemented as reusable plan structures. We call the agent that uses these plans, an adaptive replication manager. The collected data will then be processed in the feedback control mechanism of the feedback controller

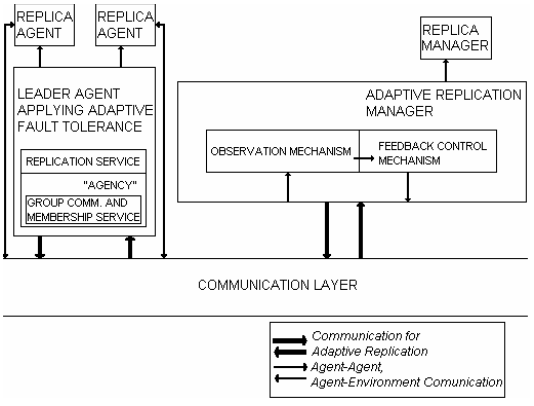


Fig. 1. An Abstract Architecture for Adaptive Replication in Fault Tolerant MAS

based adaptive replication infrastructure to compute the agent's criticality. The output of the feedback control mechanism is sent within the content of a FIPA message to the leader *(*there is a coordinator agent that controls all the replicas of itself) agent(s) applying the adaptive fault tolerance policy.

On the left part of the Fig. 1, the internal architecture of an agent applying the adaptive fault tolerance policy is shown. The necessary services to improve fault tolerance such as membership service, group communication service, and failure detector are embedded into the Agency Package (agent's internal architecture) of SEAGENT's layered architecture. The replication service is placed over the layer including the Agency Package of the SEAGENT platform. The agents pass the message received from the adaptive replication manager to the replication service implemented as a reusable plan structure. The replication service of an agent replicates new replicas or removes replicas with respect to the content of the FIPA message received from the adaptive replication manager.

3 SEAGENT Architecture

In this section, we briefly introduce SEAGENT's layered software architecture to explain how adaptive replication scheme is integrated into the SEAGENT architecture [5, 6, 7]. SEAGENT platform architecture is shown in Fig. 2. The bottom layer of the platform architecture is responsible for abstracting the platform's communication infrastructure implementation. SEAGENT implements FIPA's Agent Communication and Agent Message Transport specifications [2] to handle agent messaging.

The second layer of the SEAGENT architecture includes packages, which provide the core functionality of the platform. The first package, called the Agency, handles the internal functionality of an agent. In order to improve fault tolerance, some services such as group communication service, group membership service, and failure detector are implemented within the agency package.

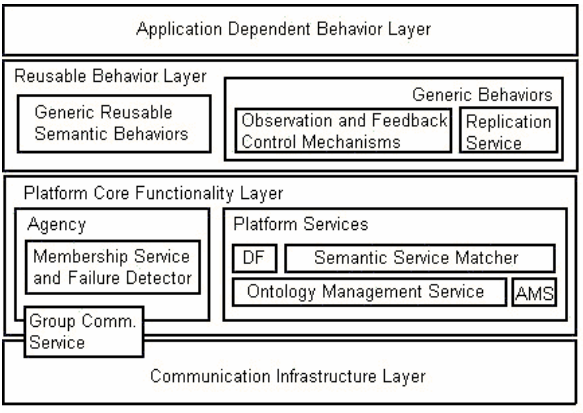


Fig. 2. SEAGENT Overall architecture

The second package of the Core Functionality Layer includes service sub-packages, one for each service of the platform such as Directory Facilitator (DF) Service and Agent Management Service (AMS).

The third layer of the overall architecture includes pre-prepared generic agent plans. We have divided these generic plans into two packages. The Generic Behavior package collects domain independent reusable behaviors that may be used by any MAS. In our approach, the replication service is implemented as reusable plans in the layer consisting of the generic behaviors plans. Thus, this makes our agents flexible in terms of fault tolerance, since it is possible to easily modify existing plans, remove some of plans, or include new plans. On the other hand, Generic Semantic Behaviors package includes only the semantic web-related behaviors.

To effectively handle faults in large-scale MAS, we implemented a feedback controller based adaptive replication infrastructure, which is responsible for dynamically adapting resource allocation and associates with the replication service. The next section describes in detail how a feedback controller based adaptive replication infrastructure is implemented.

4 A Feedback Controller Based Adaptive Replication Infrastructure

A typical feedback control system includes a controller, a plant to be controlled, actuators, and sensors. It defines a controlled variable, the quantity of the output which is measured and controlled. The set point is the correct value of the controlled variable. The difference between the current value of the controlled variable and the set point is the error. The manipulated variable is the quantity that is varied by the controller so as to affect the value of the controlled variable.

The adaptation mechanism in our approach utilizes a feedback control technique to achieve satisfactory system performance in replication. To apply feedback control in replication, we need to identify the controlled variable, the manipulated variable, the

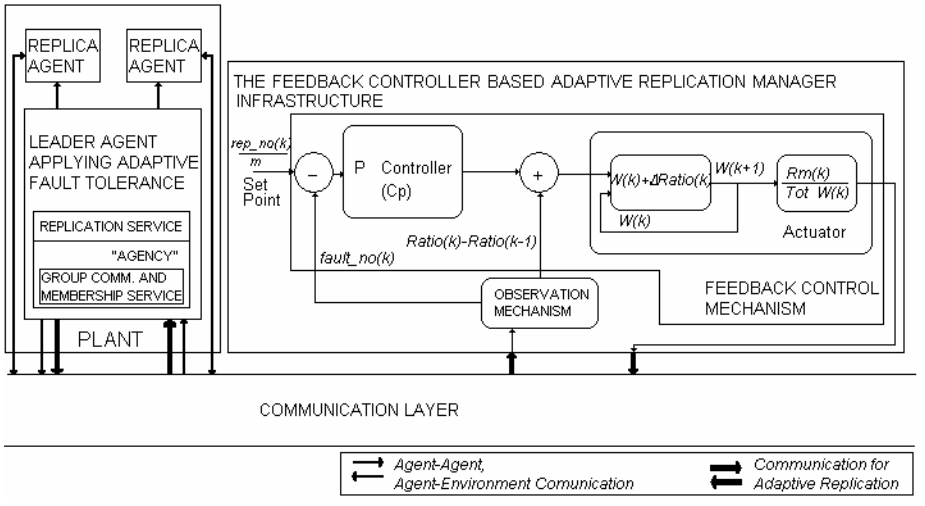


Fig. 3. The Architecture for Feedback Controller Based Adaptive Replication

set point, the error, and the control function. The number of replicas in a replica group is defined as the controlled variable in our approach. The manipulated variable is the failure rate in a replica group. The set point will be chosen by the programmer since the difference between the manipulated variable and the value of the set point (the error) has an effect on defining the replication degree. In our approach, the set point is defined as one third of the replication degree to reach agreement in asynchronous environments, as described in Section 2. Therefore if the number of failures (detected by a failure detector) exceeds one third of the replication degree, then error (the difference between the number of failures and one third of the replication degree) will increase the next value of the replication degree. The architecture for the feedback controller based adaptive replication is illustrated in Fig. 3.

We have clearly seen the plant, the actuator and the controller of the feedback controller based adaptive replication infrastructure in Fig. 3. The plant is a replica group including its leader. The observation mechanism is a designated sensor to monitor the data from the environment where agents exist. The next section describes the observation mechanism of the feedback controller based adaptive replication infrastructure.

4.1 The Observation Mechanism

Monitoring is necessary for acquiring information to determine the criticality of agents in adaptive fault tolerance policies. In our approach, monitoring is achieved via an observation mechanism. The observation mechanism is responsible for acquiring data from the system-level such as the number of messages, message sizes, and fault rates and data from the application-level, such as the role criticality of agents, in our approach. We implemented this mechanism as a reusable plan structure. An adaptive replication manager as a BDI (**B**eliefs, **D**esires and **I**ntentions) agent executes this plan to monitor the environment.

According to this plan, the adaptive replication manager queries the leader agents about their communication loads and the number of failures in its replica group. Thus, all data from system and application levels can be acquired by querying agents in a certain period. This data collection period is set during the initialization of an organization. We note that the adaptive replication manager is replicated with the passive replication strategy.

4.2 The Feedback Control Mechanism

In this section, we explain how we process data collected by the observation mechanism to obtain the criticality information of each agent. To process the collected data, we have implemented a feedback control mechanism, illustrated in Fig. 3 as a part of the calculation of agents' criticality plan of the adaptive replication manager.

In this mechanism, during system initialization, a period is set for organization. The plan related to the feedback control mechanism is executed in this period. This period is called the sampling period T and actually defined over a time window ((k-1)T, kT), where k is the sampling instant. According to the plan related to the feedback control mechanism, adaptive replication manager calculates the criticality of each agent by using the following formulas:

$$\Delta fault(k) = fault\_no(k) - replica\_no(k)/m \quad (1)$$

fault_no(k): The number of failures in a replica group which is detected by the failure detector in a replica group in the *kth* sampling window. If this data for the replica group of an agent is not larger than the set point, there is no point worrying about the system's availability.
replica_no(k): The number of replicas in a replica group in the *kth* sampling window.
m: A coefficient set by the programmer. For a Byzantine failure model, m should be equal to 3, as explained in Section 2.
$\Delta fault(k)$: The difference between *fault_no(k)* and *replica_no(k)* which is called error of the feedback control mechanism. If the value of $\Delta fault$ is larger than 0, this means that the contribution of the fault rate to the criticality should be increased. Thus, the controller is activated (Otherwise Cp is set to 0). Proportional gain of the controller is restricted by the following:

$$0 < Cp < (6 * Tot\_W(k-1) / RM) \quad (2)$$

Cp: Proportional gain of the controller. Based on the above inequality, Cp should be chosen between the bounds defined by the inequality. We derive this value from the stability analysis of the feedback control mechanism.
RM: The available resources that define the maximum number of possible simultaneous replicas.
Tot_W(k-1): Total criticality of all agents in a multi-agent system for the previous sampling period.

$$\Delta Ratio(k) = \Delta fault(k) * Cp + Ratio(k) - Ratio(k-1) \quad (3)$$

$\Delta Ratio(k)$: The change in the criticality of an agent in the *kth* sampling window.

Ratio(k): This is the degree of an agent's activity in a MAS organization at the *kth* sampling instant. The value of *Ratio(k)* is calculated using the value of the role criticality of the agent, and communication load (message size and the number of requests received) of the agent. We consider *Ratio(k)* of an agent as a numerical value within the interval [0 1]. *Ratio(k-1)* is the *Ratio* of an agent from the previous period.

The formula for the actuator of the feedback control mechanism is given below;

$$W(k)= W(k-1)+ \Delta Ratio(k) \tag{4}$$

W(k): New criticality value to be applied in the next period. *W(k-1)* is determined in the *(k-1)th* sampling period. In the next step, we determine the normalized criticality as follows:

$$Wratio(k)=W(k)/Tot\_W(k) \tag{5}$$

Wratio(k): Normalized criticality of an agent in a MAS organization in the *kth* sampling period. *Wratio(k)* shows how critical an agent is among other agents in a MAS organization. *Wratio(k)* values are sent to the agents to be used in the adaptive fault tolerance plan which is implemented as a reusable plan using HTN formalism [3].
Tot_W(k): Total criticality of all agents in a MAS organization.

In the adaptive fault tolerance policy of our approach, the leader has the "Adaptive Fault Tolerance" plan in which adaptive replication mechanism is performed. In this plan, the leader determines the replication degree of the group by using *Wratio* values sent by the adaptive replication manager. Since the resources are limited, the replication degree for each replica goup is defined as follows:

$$R.D=rounded(Wratio*RM) \tag{6}$$

where *R.D* is the replication degree of a replica group that should be present in current sampling period and *RM* is the number of available resources in an organization.

The plan structure, which adapts the replication degree of a group, is executed as explained in [5]. In order to give an idea about adaptive replication implemented using SEAGENT, we'll now present a case study.

5 Case Study

For the evaluation of our approach, we designed an agent system which includes some specific agents called library assistant agents and some other agents that are specially designed to query library assistant agents. A library assistant agent holds the library ontology for the books which exist in our department library. Instances of this ontology hold the properties of books including name, ISBN, authors' names and keywords of the books. The library assistant agent is queried by the agents with regard to a specific book. In our case study, the library assistant agent has only one plan that matches the request to the book ontology instance(s) and returns the matched books' descriptions within a FIPA message. Some agents also have simple plans that directly query the library assistant agents and present the results returned by the library assistant agent to the user interface. In this case study, the other agents depend on the library assistant agent. Therefore, the library assistant agent is a single of point

of failure. Since it is a critical agent for the system's operation, it must be initialized as a fault tolerant agent to make the system more robust.

The agent system is implemented in the SEAGENT platform and Java Version 1.5.0. The tests are performed on two computers with Intel Pentium 4 CPU running at 1.5 GHz and 2GB of RAM, running Windows XP.

In this test, we evaluated the effectiveness of our approach as the number of faults in a replica group increases up to 128. Thus, we implemented a test bed consisting of five library assistant agent leaders and five querying agents, and a failure simulator. The failure simulator sends "kill" messages to the agents within a certain period to simulate the presence of faults. The agents receiving the "kill" messages will stop their threads. Most of the kill messages are sent to the more critical agents while fewer of the kill messages are sent to the less critical agents.

In the first case, we employ the system without a feedback control mechanism by setting Cp to 0. We cancel the closed loop of the feedback control mechanism and consider only the communication load and the role of agents when calculating the criticality of agents. Thus we try to observe the behavior of a control mechanism similar to that explained in [4]. In the second case, we employ the feedback control mechanism for different values of Cp according to the formula (2). We consider the set of values of Cp such as *(3/4*(6* Tot_W(k-1) / RM))*, *(1/4*(6* Tot_W(k-1) / RM))* and *(6* Tot_W(k-1) / RM),* and *(3/2*(6* Tot_W(k-1) / RM))* for each run. In this test, the number of replicas (RM) in the organization must be 100. Thus, we try to observe the effect of the feedback control mechanism by comparing two cases.

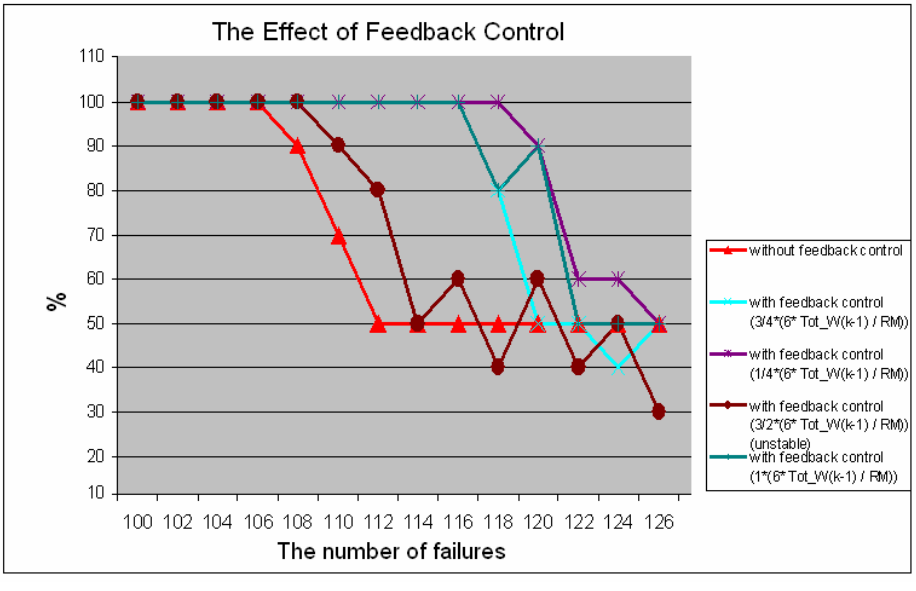


Fig. 4. The Effectiveness of the Feedback Control Mechanism

The results are illustrated in Fig. 4. As seen from the figure, as the number of the failures increases over 100, the success rate, which means the percentage of the replica groups that can complete their tasks, in the organization eventually decreases,

since only 100 replicas can be present in the organization (Rm is set to 100). This result was expected, since as the number of failures linearly increases, the feedback control mechanism has to consider the number of failures to determine the criticality values of agents. By setting the value of Cp to zero, the number of failures will not be considered in the determination of the criticality of agents. Therefore when the failure rate increases in the organization, the feedback control mechanism shows the better performance compared to the other control mechanisms without feedback control. This result proves that the integration of control theory to adaptive replication improves fault tolerance. Moreover, we experimentally prove that Cp should be chosen between the bounds defined by the inequality given by the formula (2). When Cp is not chosen within the bounds, the performance will decrease and unstable behaviors can be observed.

6 Conclusion

In this paper, a new approach has been proposed for adaptive replication based on the classical control theory. This approach is used to dynamically evaluate the criticality of the agents. The replication degrees of agent replica groups are defined with respect to the criticalities of the agents.

We know that adaptive replication is a useful technique in terms of fault tolerance. While doing experiments, we observe that our new approach to adaptive replication shows a better performance compared to control mechanisms without using feedback control. Moreover, the feedback controller based adaptive replication mechanism for multi-agent systems can survive Byzantine failures in asynchronous environments. In conclusion, the results are indicating that efficient replication is sustainable using this model.

References

1. Bracha, G., Toueg, S.: Asynchronous Consensus and Broadcast Protocols. Journal of the ACM 32(4) (1995)
2. FIPA: FIPA Specifications, http://www.fipa.org
3. Paolucci, M., et al.: A planning component for RETSINA Agents. In: Jennings, N.R. (ed.) ATAL 1999. LNCS, vol. 1757, Springer, Heidelberg (2000)
4. Guessoum, Z., Ziane, M., Faci, N.: Monitoring and organizational-level adaptation of multi-agent systems. In: AAMAS'04, pp. 514–522. ACM, New York (July 2004)
5. Bora, S., Dikenelli, O.: Implementing A Multi Agent Organization That Changes Its Fault Tolerance Policy at Run-time. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, Springer, Heidelberg (2006)
6. Bora, S., Dikenelli, O.: Applying Feedback Control in Adaptive Replication in Fault Tolerant Multi-agent Organizations. In: SELMAS'06. Proc. ICSE'06 Fifth International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, Shangai, China, ACM, New York (2006)
7. SEAGENT, http://seagent.ege.edu.tr

Exploring Social Networks in Request for Proposal Dynamic Coalition Formation Problems

Carlos Merida-Campos and Steven Willmott

Universitat Politècnica de Catalunya, Software Department, E-08034 Barcelona, Spain
{dmerida,steve}@lsi.upc.edu

Abstract. In small scale multi-agent environments, every agent is aware of all of the others. This allows agents to evaluate the potential outcomes of their interaction for each of their possible interaction partners. However, this farsighted knowledge becomes an issue in large scale systems, leading to a combinatorial explosion in evaluation and is unrealistic in communication terms. Limited awareness of other agents is therefore the only plausible scenario in many large-scale environments. This limited awareness can be modeled as a sparse social network in which agents only interact with a limited subset of agents known to them. In this paper, we explore a model of dynamic multi-agent coalition formation in which agents are connected via fixed underlying social networks that exhibit different well known structures such as *Small World*, *Random* and *Scale Free* topologies. Agents follow different exploratory policies and are distributed in the network according to a variety of metrics. The primary results of the paper are to demonstrate different positive and negative properties of each topology for the coalition formation problem. In particular we show that despite positive properties for many problems, *Small World* topologies introduce blocking factors which hinder the emergence of good coalition solutions in many configurations.

1 Introduction

In Multi-Agent Systems, agents need to interact in order to fulfil their common or individual goals. In some cases, an agent does not need to interact with every other agent, and social knowledge can be spread in the basis of some neighbourhood model such as functional neighboring (as in supply chain models [18]), or geographical neighboring (as in sensor networks [5]). In other cases, agents might benefit from being aware of every other agent of the population (*farsighted* social knowledge). This is the case of organisational systems [11,14,15] where agents have to explore a search space of agent group combinations in order to improve data flow, allocate resources efficiently, solve a problem in a coordinated manner, or to improve their outcomes by creating alliances. However, farsighted social knowledge is not possible in large scale Multi-Agent Systems, since it quickly leads to combinatorial explosion and intractability. In such scenarios, a fallback

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 143–152, 2007.

position is for agents only to interact with near neighbours initially and attempt to use these connections to find other agents if necessary.

The work presented here, covers results on the properties that specific network topologies exhibit under different exploratory behaviors for the specific problem of *Coalition Formation*, using a dynamic formation mechanism (Iterated RFP [14]). Instead of having agents that dynamically adjust their social connections [13,9] or that propagate/contaminate their neighbors through a social network [6,17], agents in the present model are constrained by a unmovable social network where no social capital is transmitted. However, agents can iteratively form coalitions with their social network's neighbors – leading to an evolutionary process that converges to a Nash equilibrium. These dynamics gives shape to coalitions by putting together agents that can be far away from each other in the social network. Results obtained show that network parameters such as node-degree distribution or clustering coefficient in concrete nodes, determine the exploratory behavior in the social network of the population. One of the main findings presented is the disadvantage of highly clustered topologies. This result coincides with what has been recently reported in [13] and in [7], where it is explained why certain human-civilisations with scattered connections between their regions, triumphed over others that were (for communication purposes) optimally connected. As with these models, the work presented here shows how random networks fosters higher exploration in the system than other optimised networks. The significant difference however is that these dynamics are seen in a coalitional scenario rather than one using one-to-one interactions.

Coalition Formation has been traditionally studied under the assumption of farsightedness and focused on the problem of finding stability concepts . Multi-Agent Systems research introduced the possibility of experimenting with coalitional systems with a limited number of possible interactions [1,16], and more recently myopic agents have been studied with concrete knowledge network topologies in team formation [9,10] as well as in firm formation models [2]. In this line, the work presented here shows results on experimentation with different underlying social network topologies on an specific type of electronic market allocation mechanism called Iterated Request For Proposal (RFP from now on). This model was first studied in [12], and further explored in [14] and [15]. In this environment, an entity regularly issues a call for tender to provide specific goods or services with certain characteristics. Providers compete amongst themselves in consortia – *coalitions*. Bids are ranked according to an evaluation of their aggregation of skills for the task, and receive a payoff according to their placement in the ranking. Structures created are based on complementarity of their members. The more complementary they are, the better outcome they obtain, assuming this way a social structure with proved real properties reported in [4]. There are many existent real systems that follow the RFP type procedures such as public building projects, competitive tender for government contracts or even collaborative research project grants.

Section 2 presents a formalisation of the RFP mechanism. Section 3 describes an specific metric that records the amount of dynamism that a certain topology

generates as well as other metrics used to perform the analysis and the experimental setup. Results are analysed in section 3. Apart from examining the drawbacks of the Small World topology, section 3.2 analyses results on the importance of positioning of agents with specific individual properties (*versatility* and *competitiveness*) in certain parts of the social network. Finally, section 4 discusses the results summarising the main conclusions and explaining future lines of work.

2 Iterative RFP Coalition Formation Model

A population $I = \{1, 2, \ldots, n\}$ consists of a finite number of n individuals or *agents*. Agents compete for creating the best solution proposal to a given task *task* T. A *partition* $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_p\}$ of the population I is a specification of p *coalitions* of agents $\sigma_i = \{\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{im}\}$, where σ_{ij} represents an agent from population I forming part of coalition σ_i. By forming coalitions, agents are able to increase their competitiveness towards the specified task. Agents have heterogeneous capabilities, thus having different performance levels in different skills. A finite number of k *skills*, indexed from 1 to k is set for which each agent σ_{ij} has a fixed value: $\sigma_{ij} = \langle \sigma_{ij}^1, \sigma_{ij}^2, \ldots, \sigma_{ij}^k \rangle$. This way, it is possible to define a continuum of possibilities between agents that are *specialised* in the performance of a certain skill being unskilled for the rest of them, and agents that are *versatile*, being averagely apt for the performance of all the skills defined. A Task T is specified by a set of k skill requirements. Each one of the k skills have a degree of requirement. These requirements are modelled in the form of a number $T = \langle T^1, T^2, \ldots, T^k \rangle$. In a coalition, skills of agents are aggregated in such a way that each agent gives the best of itself in a join effort to create a group as competitive as possible under the requirements of the Task. The coalition has a value in each skill representing the aggregated effort of its members. The aggregation for every skill $l : 1 \leq l \leq k$ in the coalition is modelled in the following way:

$$\sigma_i^l = \max(\sigma_{ij}^l) : 1 \leq j \leq m \tag{1}$$

Each skill is considered as a necessary subtask for performing task T. By using the aggregation function shown in equation 1, the agent in a coalition which is the best fit for performing a certain subtask will be the one that performs it. The aggregated effort of agents in equation 1 is used to measure an score $scr(\sigma_x, T)$ that tells how well the agents in coalition σ_x perform together for accomplishing a task specification T. The score of a coalition is computed as the scalar product between σ_i and T:

$$scr(\sigma_i, T) = \sum_{l=0}^{k} (\sigma_i^l * T^l) \tag{2}$$

Agent Choices and Strategies: Each player's strategic variables are its coalition choice to join σ_j and a set of agents in this coalition to eliminate: $\phi_{jk} : \{(\phi_{jk} \subset \sigma_j) \vee (\phi_{jk} = \emptyset)\}$. The possibility for optimisation responds to the

change of value that certain agents can experiment when in their coalition they are out-skilled by a new member and so they become redundant. Only those actions accepted by a majority (more than the half) of members in the affected coalition, are performed. An agent that is requested to take an action can submit a finite number of requests in an specific order, in such a way that if an action is not accepted, the next action is checked and so on. If none of its action proposals is accepted, the agent stays in the same coalition where it was.

All the agents in the population follow the *Competitive Strategy* [14], that consists on proposing a set of actions that contain every proposal that either improves the score of the coalition the agent is in, or keeps the same score while reducing the size of the coalition. When they receive a proposal from an outsider, they accept if they are not in ϕ_{jk}'s proposal, and if the proposal improves the score or keeps it while reduce the coalition size. Every agent j has a fixed social network α_j that is a non-empty set of agents. When considering to join a different coalition, agents are limited to just evaluating coalitions of agents in α_j.

RFP Iterated Model: At time 0, every agent is a coalition of just one element ($\sigma_i = \{\sigma_{ii}\}$). A task T is issued and a run of negotiation starts in which every agent, sequentially and following a random order, is asked about an action to take (see previous subsection). Agents have no knowledge on the order in which they are requested for an action, and when they are asked they can consider any action involving the coalitions in which there is some member of their social network. The run ends when all agents have been requested for an action. The process last as many runs as necessary until it converges to an stable state. Stability is reached when in a complete run, no agent is willing to leave the coalition is in or none of its actions are accepted by the hosting coalition.[1] When the system has reached equilibrium, coalitions' scores are captured for further analysis.

3 Experiments

Metrics used in the experiments: In order to measure how competitive an agent is in a simple way, every agent in the population has the same total number of skill capabilities, but distributed differently across skills. This way we can define a simple metric of *Competitiveness*: $com(\sigma_{ij})$, by just measuring the standard deviation in its skill values weighted by the task values. Analogously, *Versatility* is defined as the inverse of *Competitiveness*: $(ver(\sigma_{ij}) = 1/com(\sigma_{ij}))$.

In order to test the dynamics that a certain underlying social network permits, a new metric called *Historical Average Degree* (HAD from now on) is defined. This metric measures the distance that exists in the social network between agents in a coalition. When an agent A joins a coalition there is at least 1 member B with HAD equal to 1, otherwise this agent would not directly know any of the members of the coalition, hence it could not consider that coalition. The distance to each other coalition member C corresponds to 1 (distance from

[1] In [15] it was shown how the system always converge to an stable state when the population follows an score maximizing function.

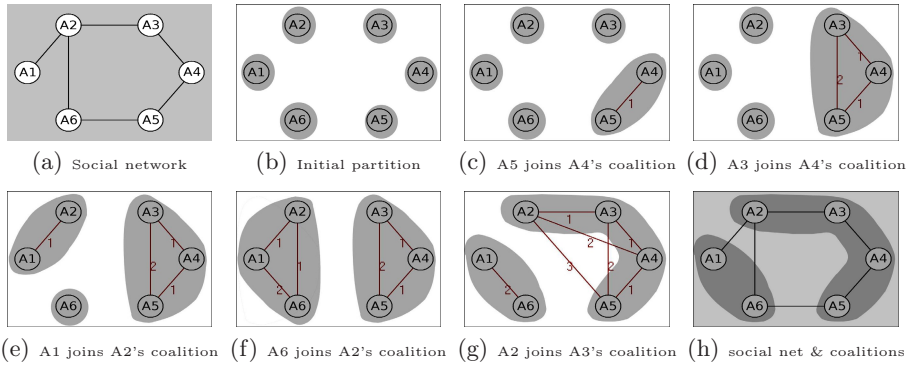


Fig. 1. Iterated coalition formation example. Edge's values reflect the HAD between nodes, shaded areas denote coalitions. Note that A2-A5 HAD is higher than its distance.

A to B) plus the minimal path length from B to C through a complete HAD valued network between coalition's members. Coalition's HAD value is the mean of each HAD value in the coalition. This value is computed for every coalition when the system has reached equilibrium. Figure1 exemplifies the computation of HAD values.

At the end of an experiment, coalitions are compared with the optimal coalitional structure σ^+; that is the structure whose coalitions have the maximum possible score[2]. If top coalitions of an experiment have low score versus those in the optimal structure, this is because the agents best able to contribute to good coalitions are stuck in suboptimal coalitions. This way, a certain experiment ends up with better or worse results depending on how efficient the best agents are in getting to know each other. To measure this efficiency, the aggregated score of an experiment is obtained by computing the weighted sum of every coalition with an exponential weight: $Sc(\sigma) = \sum_{\text{rank}=0}^{p}(\text{scr}(\sigma_{\text{rank}}, T) * 1/2^{\text{rank}})$. To know how suboptimal a coalitional setup is the following difference is computed: $\text{Sub} = \text{Sc}(\sigma^+) - \text{Sc}(\sigma)$.

3.1 Experimental Set-Up

In order to investigate the effects of network properties in the dynamics of exploration of agents, a significant number of experiments have been performed varying the most significant variables. These are the social network topology and the mapping between agents and nodes in the network. A total of 4000 networks with 500 nodes and average undirected connectivity degree of $k = 10.3$ have been tested, in 4 different topologies: 1000 *Small World* (SW) networks (using the Watts-Strogatz model [19] with $p = 0.07$), 1000 *Random* (Rdm) networks (Using Ërdos-Reni model [8]), 1000 *Random2* (Rdm2) networks (Using

[2] This reference structure is computed by exhaustive exploration using farsighted agents (agents with a complete social network).

Table 1. Average properties across the set of 1000 networks used in each topology

	node degree (k)	avg. distance (ad)	clust. coefficient (cc)
SF (Scale Free)	10.3	2.571	0.072
Rdm (Random Ërdos)	10.3	2.644	0.028
Rdm2 (Random Watts)	10.3	2.643	0.024
SW (Small World)	10.3	3.541	0.565

Watts-Strogatz model with $p = 0.07$ and randomly rewiring the nodes while keeping every node's degree the same) and 1000 *Scale Free* (SF) networks (using Barabassi model [3]). Scale Free networks have a characteristic degree distribution by which a minority of nodes have high connectivity degree while the majority is poorly connected, this is a recurrent topology in many macro/micro-economic and social distributions amongst many other scenarios. Small world is characterised by having a high clustering coefficient between nodes and short average path lengths, and is a characteristic property of the internet structure and human contact networks amongst many other networked structures.

For each one of the networks, 3 different mappings have been created in the following way: in one hand, for every network, nodes are ordered decreasingly by connectivity degree k. On the other hand, 2 different orders of agents are created by computing *com*, and *ver* metrics in every agent (see section 3) and ordering agents decreasingly by each metric's values. Every experiment maps every agent in one of the 2 specific orders (*ver-ord* or *com-ord*), to a node according to its k order in the network. An additional random mapping *rdm-ord* has been used to create the third tested order. That creates a total setup of 12000 experiments: for each 1000 networks of each of the 4 topologies tested, 3 different sets of experiments are performed, each one with a concrete mapping between node degree and agent characteristics.

For space restriction reasons, not all the experiments performed are shown. Some of the variables have been fixed. These variables are: the population composition (500 agents with 10 skills and, for every agent a total value of 100, heterogeneously distributed amongst the skills. The stdev. in the skill distribution is homogeneously distributed from 5 to 20), the task used in all the experiments ($T = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$ [3]) and the connectivity degree k (10.3 for each one of the 4000 networks).

3.2 Experiments Results

Experiments with rdm-ord mapping: This setup does not prioritise any specific type of agent in the experiments this way the effect of the topology under even conditions for every agent can be observed. The average suboptimality of each topology is summarised in Figure 2. There is a clear disadvantage in the

[3] the difference between values favors the diversity of Competitiveness and Versatility degrees (see section 3). None of the values is 0, so that all the skills are required (see section 2).

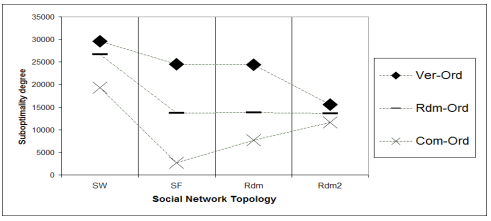
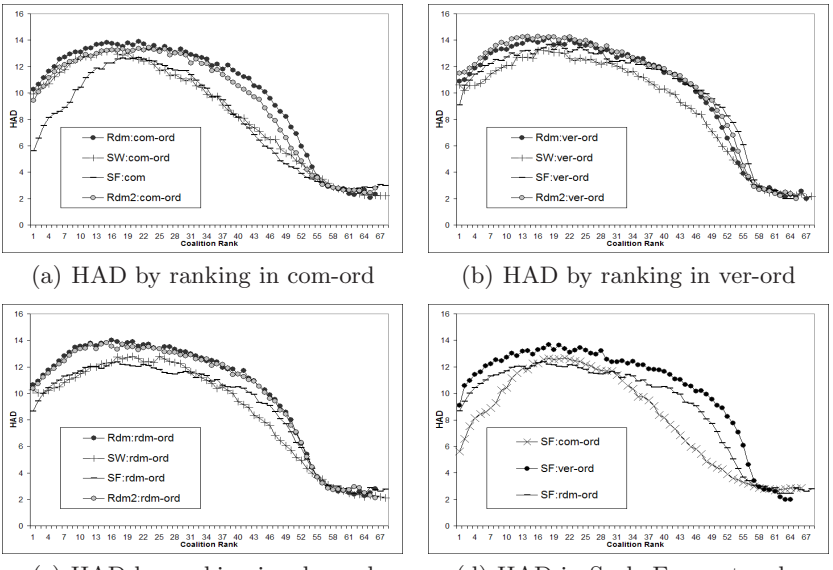


Fig. 2. Suboptimality for each topology and mapping. Topologies with higher node degree heterogeneity show higher sensitivity to specific agent mappings. SW proves to be the most suboptimal topology.

SW networks. These networks fail to provide the necessary conditions to put in contact the most competent agents. There are two main reasons that explain this fact. First, the highest average distance in the nodes of the SW networks (see Table 1), is an indicator as to how short the path that interconnects two nodes can be. The longer this path is, the higher can be the number of required jumps in the network to group two agents. However this indicator is not the only fact that explains the suboptimality of SW networks, as this metric accounts for the average shortest distance, but agents do not necessarily follow the shortest path that interconnects them. Paths can be blocked when intermediate agents have an stable coalition that does not accept new members. In such situations, in order to join specific agents, a network topology must provide not only *optimal paths* but also as many *alternative paths* as possible. The structured nature of SW networks involves that, initially, agents form coalitions with other agents within their cluster; coalitions iteratively are expanded because structured clusters are interconnected. This raises HAD in such coalitions to similar levels than the SF networks (see figure 3(c)) but the process of exploration is heavily dependant on agents with high *betweenness* (rewired agents in the Watts model) – those that create shortcuts between clusters. If these agents are stuck in a certain coalition, they are blocking the path that interconnects distant clusters, hence forcing agents to explore via neighbor clusters.

Another interesting fact captured in Figure 3(c) is that Rdm and Rdm2 networks show higher average HAD compared to SF. However, SF networks prove to have similar performance than Rdm and Rdm2 (see Figure 2). Again, the average distances partly explains this fact; as SF networks have, on average, shorter optimal distances, some agents can find the way to each other in a shorter number of steps than for the case of Rdm and Rdm2 networks. However there is a structural reason than makes SF networks create coalitions with less HAD. As in the case of SW networks, some nodes with a crucial network role might block the process. While for SW networks these were the shortcut agents, in SF, these are the agents with higher degree (hubs). When these agents form an stable coalition, all the connections that were concentrated in them are lost and all the paths that crosses them are blocked, so the exploration is affected by an heterogeneous degree distribution when the stabilisation is early reached.



(a) HAD by ranking in com-ord

(b) HAD by ranking in ver-ord

(c) HAD by ranking in rdm-ord

(d) HAD in Scale Free networks

Fig. 3. Avg. HAD of the 67 best coalitions through 1000 experiments showing the amount of exploration that each topology permits

This way, the advantage of having a shorter avg. distance is compensated by the disadvantage of having nodes accumulating many connections.

Experiments with ver-ord and com-ord mapping: Network structures with highly heterogeneous nodes degree are more affected by a specific mapping. The SF topology is the one with the most unequal distribution, followed by Rdm. In SW and Rdm2, the degree distribution in nodes is almost even, and so these are not very sensitive to the different mappings.

An interesting result drawn from these experiments shows that when the more competitive agents (those that are ment to be part of the leading coalitions) have the lowest number of connections in the network, the general exploration performed by agents rises significantly (ver-ord fosters higher HAD than other mapping, see Figure 3(d)). This happens because competitive agents have a higher potential of attraction. As these agents are valuable parts of many possible coalitions, others are attracted to join them. This attraction fosters the mobility of many agents that improve the score of the coalition of the competent agent by attracting more competent agents each time. However, in many cases, this incremental attraction proves not to be enough in order to let the most competitive agents to get to know each other. This is why ver-ord experiments produce the highest suboptimality degree (Figure 2).

Inversely, when competitive agents are highly connected, good quality coalitions are formed in a few jumps, having near optimal scores. Their members are highly connected hubs in the social network that easily find a path to get in

touch. However, in the lower ranking positions, the HAD analysis shows insufficient dynamics to generate good results because there are not many attractive agents to foster mobility between coalitions. In this case, the coalitions at lower positions end up having poor performance since they are limited to exploring little further than their close neighbors (Figure 3(d) in the SF:com-ord series).

4 Conclusions and Future Work

In the rush to connectedness and the formation of structured organisational topologies, the recent work by [13] suggested a potential downside of highly efficient topologies with short average path lengths. In this work, we suggest that there are indeed arguments in favor of unstructured networks – and specifically in scenarios which use not one-on-one interactions but coalitional interactions. As in [13], The Request for Proposal model also shows that structured networks reduce the exploration possibilities. When highly compatible agents are closely connected in the social network, diversity in the system is squeezed out and agents are unable to find out optimal solutions. Additionally, the RFP model suggests that Small World networks are potentially inefficient because they concentrate on a reduced set of agents to provide key connections between different clusters. When those agents block the exploration, the system end up having worse global results. Another similar blocking factor is demonstrated in Scale Free networks for which overall results are heavily dependant on highly connected agents.

In spite of the attention that dynamic coalition formation has attracted in recent years, the use of social networks to map limited awareness of agents has not been deeply explored (for an exception see [10,9] in the area of team formation). In this work we aim to take a step further in the combination of social network analysis and coalition formation, showing how the dynamics of coalitional systems are affected by the topological structures of underlying social networks and also highlighting properties of well known network topologies which are not beneficial in all cases. The work in progress is focused on testing the effects of placing specific agents in nodes of the network with other network properties such as page-rank centrality, betweenness or clustering coefficient.

References

1. Axtell, R.: The emergence of firms in a population of agents: Local increasing returns, unstable nash equilibria, and power law size distributions. Working Paper 3, Center on Social and Economic Dynamics, Brookings Institution (1999)
2. Axtell, R.: Effects of interaction topology and activation regime in several multi-agent systems. In: Moss, S., Davidsson, P. (eds.) MABS 2000. LNCS (LNAI), vol. 1979, pp. 33–48. Springer, Heidelberg (2001)
3. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
4. Burt, R.: Structural Holes: The Social Structure of Competition. Belknap Press (1995)

5. Culler, D., Estrin, D., Srivastava, M.: Guest editors' introduction: Overview of sensor networks. Computer 37(8), 41–49 (2004)
6. Delgado, J., Pujol, J.M., Sanguesa, R.: Emergence of coordination in scale-free networks. Web Intelligence and Agent Systems 1(2), 131–138 (2003)
7. Diamond, J.: Guns, Germs, and Steel: The Fate of Human Societies. W.W. Norton and Co., New York (1999)
8. Erdos, P., Renyi, A.: On random graphs i. Publicationes Mathematicae Debrecen 6, 190–297 (1959)
9. Gaston, M.E., desJardins, M.: Agent-organized networks for dynamic team formation. In: Proceedings of AAMAS '05, pp. 230–237. ACM Press, New York (2005)
10. Gaston, M.E., Simmons, J., desJardins, M.: Adapting network structures for efficient team formation. In: Proceedings of AAMAS-04, Workshop on Learning and Evolution in Agent-based Systems (2004)
11. Horling, B., Lesser, V.: A Survey of Multi-Agent Organizational Paradigms. The Knowledge Engineering Review 19(4), 281–316 (2005)
12. Kraus, S., Shehory, O., Tasse, G.: Coalition formation with uncertain heterogeneous information. In: Proceedings of AAMAS'03, Melbourne, Australia, pp. 1–8 (2003)
13. Lazer, D., Friedman, A.: The hare and the tortoise: the network structure of exploration and exploitation. In: dg.o2005: Proceedings of the 2005 national conference on Digital government research, Digital Government Research Center, pp. 253–254 (2005)
14. Merida-Campos, C., Willmott, S.: Agent compatibility and coalition formation: Investigating two interacting negotiation strategies. In: Proceedings of TADA/AMEC Workshop, Hakodate, Japan (2006)
15. Merida-Campos, C., Willmott, S.: The effect of heterogeneity on coalition formation in iterated request for proposal scenarios. In: Proceedings of the EUMAS Workshop, Lisbon (2006)
16. Milchtaich, I., Winter, E.: Stability and Segregation in Group Formation. Games and Economic Behaviour 38, 318–346 (2001)
17. Pujol, J.M., Flache, A., Delgado, J., Sangüesa, R.: How can social networks ever become complex? modelling the emergence of complex networks from local social exchanges. Journal of Artificial Societies and Social Simulation 8(4), 12 (2005)
18. Thadakamalla, H.P., Raghavan, U.N., Kumara, S., Albert, A.: Survivability of multiagent-based supply networks: A topological perspective. IEEE Intelligent Systems 19(5), 24–31 (2004)
19. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature (393), 440–442 (1998)

Formalizing Context-Based Behavioural Compatibility and Substitutability for Role Components in MAS

Nabil Hameurlain

LIUPPA Laboratory, Avenue de l'Université, BP 1155, 64013 Pau, France
nabil.hameurlain@univ-pau.fr

Abstract. In this paper we focus on a new approach for the definition of context-based compatibility and substitutability of roles in MAS, and provide a formal framework for modeling roles together with their composition. First, we introduce the concept of usability of roles, and based on that we define two flexible roles compatibility relations depending on the context (environment). The proposed compatibility relations take into account the property preservation such as the completion and the proper termination of roles. Then, our formal framework is enhanced with the definition of two flexible behavioral subtyping relations related to the principle of substitutability. Finally, we show the existing link between compatibility and substitutability of roles, namely the preservation of the proposed compatibility relations by substitutability.

1 Introduction

In Multi-Agent Systems (MAS), roles are basic buildings blocks for defining the behavior of agents and the requirements on their interactions. Modeling interactions by roles allows a separation of concerns by distinguishing the agent-level and system-level concerns with regard to interactions [2, 3, 5, 11]. Usually, it is valuable to reuse roles previously defined for similar applications, especially when the structure of interaction is complex. To this end, roles must be specified and validated in an appropriate way, since the composition of independently developed roles can lead to the emergence of unexpected interaction among the agents.

In our previous work [5], we have shown that the facilities brought by the Component Based Development (CBD) approach [10] fit well the issues raised by the use of roles in MAS, and defined RICO (Role-based Interactions COmponents) model for specifying complex interactions based on the roles. RICO proposes a specific definition of role, which is not in contrast with the previously approaches exploiting the concept of role [2, 3, 11], but is quite simple and can be exploited in specifications and implementations. In RICO model, when an agent intends to take a role, it creates a new component (i.e. an instance of the component type corresponding to this role) and this role-component is linked to its base-agent. Then, the role is enacted by the role-component and it interacts with the role-components of the other agents. In [6], a Petri-net based formal specification for RICO is given together with the semantics for the compatibility and substitutability of roles.

In this paper, we focus on a new approach to the definition of more flexible compatibility and substitutability relations for roles which depend on the context (environment).

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 153 – 162, 2007.

The proposed approach uses the concept of *usability* of roles and then provides a formal framework for modeling usable role-components and their composition. This paper extends the work presented in [6, 7] as follows: (a) it proposes a new composition operator between roles allowing to assembly many instances of a same role, (b) it provides a new approach to the definition of more flexible compatibility and substitutability relations for roles, (c) it makes the existing link between compatibility and substitutability, and namely the preservation of the proposed compatibility relations by substitutability relations, which seems to be necessary when we deal with incremental design of role-based complex interactions.

2 Role-Based Interactions Components Modeling

2.1 The Component-Nets Formalism (C-Nets)

Backgrounds on labelled Petri nets. A marked Petri net $N = (P, T, W, M_N)$ consists of a finite set P of places, a finite set T of transitions where $P \cap T = \varnothing$, a weighting function $W : P \times T \cup T \times P \rightarrow \mathbb{N}$, and $M_N : P \longrightarrow \mathbb{N}$ is an initial marking. A transition $t \in T$ is enabled under a marking M, noted M (t >, if $W(p, t) \leq M(p)$, for each place p. In this case t may occur, and its occurrence yields the follower marking M', where $M'(p) = M(p) - W(p, t) + W(t, p)$, noted M(t> M'. The enabling and the occurrence of a sequence of transitions $\sigma \in T^*$ are defined inductively. The preset of a node $x \in P \cup T$ is defined as ${}^\bullet x = \{y \in P \cup T, W(y, x) \neq 0\}$, and the postset of $x \in P \cup T$ is defined as $x^\bullet = \{y \in P \cup T, W(x, y) \neq 0\}$. Let A be a set of methods, that is the alphabet of observable actions, and $\{\lambda\}$ denotes special unobservable actions. The symbol λ plays the usual role of an internal action. We denote as $LN = (P, T, W, M_N, l)$ the (marked, labelled) Petri net whose events represent actions, which can be observable. It consists of a marked Petri net $N = (P, T, W, M_N)$ with a labelling function $l: T \longrightarrow A \cup \{\lambda\}$. Let ε be the empty sequence of transitions, l is extended to an homomorphism $l^*: T^* \longrightarrow A^* \cup \{\lambda\}$ in the following way: $l(\varepsilon) = \lambda$ where ε is the empty string of T^*, and $l^*(\sigma.t) = l^*(\sigma)$ if $l(t) \in \{\lambda\}$, $l^*(\sigma.t) = l^*(\sigma).l(t)$ if $l(t) \notin \{\lambda\}$. In the following, we denote l^* by l, LN by (N, l), and if $LN = (P, T, W, M_N, l)$ is a Petri net and l' is another labelling function of N, (N, l') denotes the Petri net (P, T, W, M_N, l'), that is N provided with the labelling l'. A sequence of actions $w \in A^* \cup \{\lambda\}$ is enabled under the marking M and its occurrence yields a marking M', noted M(w>> M', iff either M = M' and $w = \lambda$ or there exists some sequence $\sigma \in T^*$ such that $l(\sigma) = w$ and $M(\sigma> M'$. The first condition accounts for the fact that λ is the label image of the empty sequence of transitions. For a marking M, Reach (N, M) = {M'; $\exists\ \sigma \in T^*$; $M(\sigma> M'$} is the set of reachable markings of the net N from the marking M.

Components nets (C-nets). The Component-nets formalism [6] combines Petri nets with the component-based approach. Semantically, a Component-net involves two special places: the first one is the input place for instance creation of the component, and the second one is the output place for instance completion of the component. A C-net (as a server) makes some services available to the nets and is capable of rendering these services. Each offered service is associated to one or several transitions that may be requested by C-nets. A service is available when one of these transitions, called *accept-*

transitions, is enabled. Besides it can request (as a client) services from other C-net transitions, called *request-transitions*, and needs these requests to be fulfilled. These requirements allow to focus either on the server side of a C-net or on its client side.

Definition 2.1 (C-net). Let CN = (P ∪ {I, O}, T, W, M_N, l_{Prov}, l_{Req}) be a labelled Petri net. CN is a *Component-net* (C-net) if and only if:
1. The labelling of transitions consists of two labelling functions l_{Prov} and l_{Req}, such that: l_{Prov} : T ⟶ Prov ∪ {λ}, where Prov ⊆ A is the set of provided services, and l_{Req} : T ⟶ Req ∪ {λ}, where Req ⊆ A is the set of required services.
2. *Instance creation*: the set of places contains a specific *Input* place I, such that $^\bullet I = \emptyset$,
3. *Instance completion*: the set of places contains a specific *Output* place O, such that $O^\bullet = \emptyset$.

Notation. We denote by [I] and [O], which are considered as bags, the markings of the Input and the Output place of CN, and by Reach (CN, [I]), the set of reachable markings of the component-net CN obtained from its initial marking M_N within one token in its Input place I. Besides, when we deal with the graphical representation of the C-nets, we use ! and ? for the usual sending (required) and receiving (provided) services together with the labeling function l instead of the two labeling functions l_{Prov} and l_{Req}.

Definition 2.2 (soundness). Let CN = (P ∪ {I, O}, T, W, M_N, l) be a *Component-net* (C-net). CN is said to be *sound* iff the following conditions are satisfied:
1. Completion option: ∀ M ∈ Reach(CN, [I]), [O] ∈ Reach(CN, M).
2. Reliability option: ∀ M ∈ Reach(CN, [I]), M ≥ [O] implies M = [O].

Completion option states that, if starting from the initial state, i.e. the activation of the C-net , it is always possible to reach the marking with one token in the output place O. Reliability option states that the moment a token is put in the output place O corresponds to the termination of a C-net without leaving dangling references.

Composition of C-nets. The parallel composition of C-nets, noted ⊕ : C-net × C-net ⟶ C-net, is made by communication places allowing interactions through observable services in asynchronous way. Given a client C-net and a server C-net, it consists in connecting, through the communication places, the request and the accept transitions having the same service names: for each service name, we add one communication-place for receiving the requests/replies of this service. Then, all the accept-transitions labeled with the same service name are provided with the same communication-place, and the client C-net is connected with the server C-net through these communication places by an arc from each request-transition towards the suitable communication-place, and an arc from the suitable communication-place towards each accept-transition. In order to achieve a syntactically correct compound C-net C = A ⊕ B, it is necessary to add new components for initialization and termination: two new places (an Input and Output places), noted {Ic, Oc}, and two new not observable transitions, noted {ti, to}, for interconnecting the input place {Ic} to the original two input places via the first new transition {ti}, and the two original output places to the output place {Oc} via the second new transition {to}. Thus, the composition of two C-nets is also a C-net, and this composition is commutative and associative.

2.2 Specification of Role-Based Interactions as Components

In our RICO model [5], a role component is considered as a component providing a set of interface elements (either attributes or operations, which are provided orrequired features necessary to accomplish the role's tasks), a behavior (interface elements semantics), and properties (proved to be satisfied by the behavior). In this paper, since we are interested in behavioural compatibility and substitutability of roles, when we deal with role components, we will consider only their behavioural interfaces that is the set of (provided and required) services together with the behaviours.

Definition 2.3 (Role Component). A Role Component for a role $\Re$, noted RC, is a 2-tuple RC = (Behav, Serv), where,

- Behav is a C-net describing the life-cycle of the role $\Re$.
- Serv is an interface, a set of public elements, through which RC interacts with other role components, for instance messaging interface. It is a pair (Req, Prov), where Req is a set of required services, and Prov is the set of provided services by RC, and more precisely by Behav.

Since the life-cycle of roles is specified by C-nets, we say that: (1) the role component satisfies the completion option if and only if its underlying C-net satisfies the completion option and (2) the role component terminates successfully if and only if its underlying C-net is sound.

Definition 2.4 (Role Components composition). A Role (Component), RC = (Behav, Serv), can be composed from a set of Role-Components RC_i = ($Behav_i$, $Serv_i$), i = 1, …, n, noted RC = $RC_1 \otimes \ldots \otimes RC_n$, as follows:

- Behav = $Behav_1 \oplus \ldots \oplus Behav_n$.
- Serv = (Req, Prov) such that Req = $\cup$ Req_i, and Prov = $\cup Prov_i$, i=1, …, n.

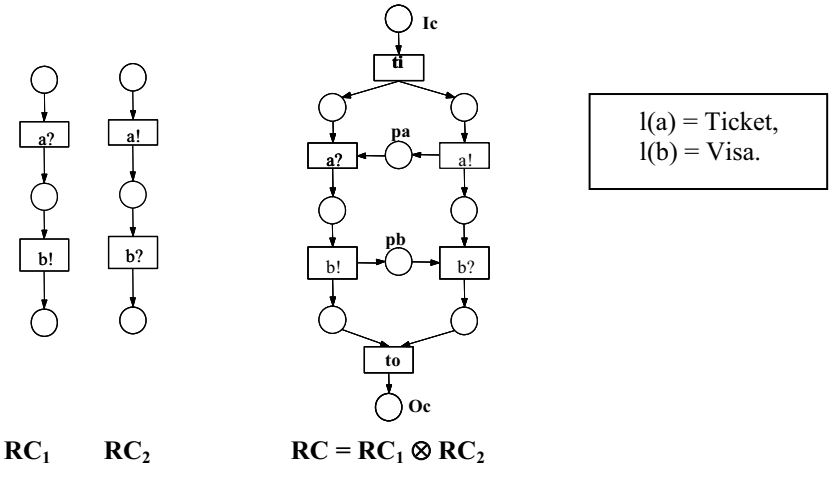


Fig. 1. RC_1 = ($Behav_1$, $Serv_1$), RC_2 = ($Behav_2$, $Serv_2$), RC = $RC_1 \otimes RC_2$, where $Serv_1$ = ({Visa}, {Ticket}), $Serv_2$ = ({Ticket}, {Visa})

The result of composing two role-components is itself a role-component, and this composition is commutative and associative. Besides, Let us notice that the composition operator ⊗ allows to assembly many instances of a same role, for instance, one instance of the buyer and many instances of the sellers in an auction protocol [5].

Example 1. Let's take the example of the ticket service and the customer. Figure 1 shows RC_1 representing the behaviour of the customer, and RC_2 the behaviour of the Ticket-service. The Ticket service initiates the communication by sending one `Ticket` and waits of the payment (`Visa`). By receiving the `Ticket`, the customer makes the payment of the ticket by `Visa`. The result of the composition of the behaviours of RC_1 and RC_2, noted $RC = RC_1 \otimes RC_2$, is shown in figure 1[1]. These two roles as well as the composed role terminate successfully.

3 Flexible Behavioural Compatibility for Roles

In component-based software engineering, classical approaches for components compatibility deal with components composition together with their property preservation [1]. In our previous work [6], we have used this approach for role-based interactions components and propose two compatibility relations. The first one deals with the correctness of the composition of roles when reasoning about the completion option, and the second deals with the proper (or successful) termination of the composed role. In this paper, we will consider explicitly the context of use of roles that is their environment in the definition of the compatibility relations for roles. The proposed approach is optimistic [1] since it allows to provide flexible compatibility relations for roles. First, let define the notion of the environment of a role.

Definition 3.1 (Environment). Let $RC_1 = (Behav_1, Serv_1)$ and $RC_2 = (Behav_2, Serv_2)$, be two role components such that $Serv_i = (Req_i, Prov_i)$, i=1, 2.
RC_2 is an environment of RC_1, and vice versa, iff $Req_1 = Prov_2$, $Req_2 = Prov_1$.
We let ENV(RC) the set of environments of the role component RC.

Definition 3.1 expresses that for two role components $RC_1 = (Behav_1, Serv_1)$ and $RC_2 = (Behav_2, Serv_2)$ such that both sets of interfaces $Serv_1$ and $Serv_2$ completely match, i.e. $Req_1 = Prov_2$ and $Req_2 = Prov_1$, the role component RC_1 is considered as an *environment* of RC_2 - and vice versa.

Given a role-component and its environment, it is possible to reason about the completion and the proper termination of their composition. Based on that, we define two notions of usability:

Definition 3.2 (Weak and Strong usability)
1. RC is weakly usable iff ∃ Env ∈ ENV(RC), Env ⊗ RC satisfies the completion option. We say that Env *weakly utilizes* RC.
2. RC is strongly usable iff ∃ Env ∈ ENV(RC), Env ⊗ RC terminates successfully. We say that Env *strongly utilizes* RC.

[1] The names of transitions are drawn into the box..

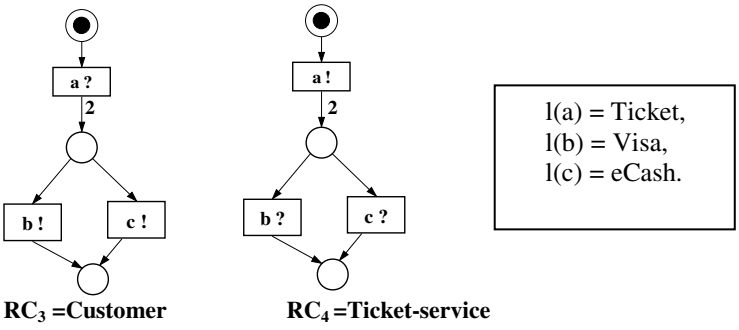


Fig. 2. RC_3 weakly utilizes RC_4, where $Serv_3$ = ({Visa, eCash}, {Ticket}), $Serv_4$ = ({Ticket}, {Visa, eCash}).

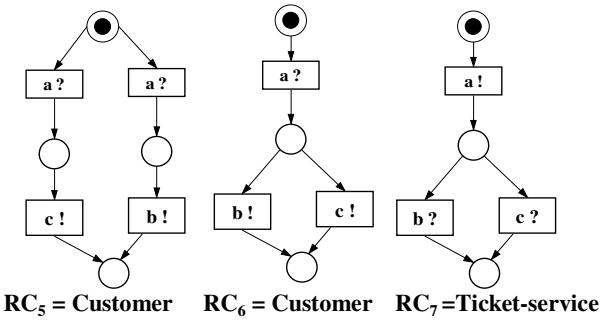


Fig. 3. RC_5 strongly utilizes RC_7, RC_6 strongly utilizes RC_7, where $Serv_5$ = $Serv_6$ = ({Visa, eCash}, {Ticket}) and $Serv_7$ = ({Ticket}, {Visa, eCash})

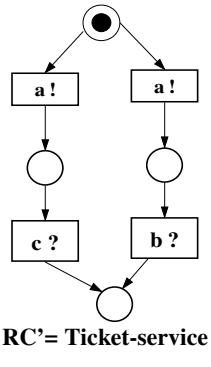


Fig. 4. RC' is not weakly usable, where Serv' = ({Ticket}, {Visa, eCash})

Example 2. Let us take again the example of the ticket service and the customer. Now, the Ticket service initiates the communication by sending (two) `Tickets` and waits of their payment (`Visa` and/or `eCash`). Figure 2 shows RC_3 representing the behaviour of the customer, and RC_4 representing the behaviour of the Ticket-service. By receiving the `Tickets`, the customer determines the kind of payment of these

two tickets. It is easy to prove that roles RC_3 and RC_4 are weakly usable, since RC_3 weakly utilizes RC_4 and vice versa. The role RC_3 is not strongly usable, since its unique weakly usable environment is the role RC_4, and $RC_3 \otimes RC_4$ satisfies the completion option but does not terminate successfully.

Example 3. In figure 3, the ticket service RC_5 initiates the communication by sending one `Ticket` and waits of the payment (either `Visa` or `eCash`). The role RC_5 and RC_7 are then two examples of the customer's behaviour. By receiving the `Ticket`, they solve an internal conflict and determine the kind of payment. The roles RC_5 and RC_7 (resp. RC_6 and RC_7) are strongly usable, since for instance RC_5 strongly utilizes RC_7 (resp. RC_6 strongly utilizes RC_7) and vice versa. However, let us take the Ticket service RC' shown in figure 4. RC' is not weakly usable since there is no environment that can weakly utilize it. Indeed, roles RC_5 and RC_6 are the two possible environments of RC' (according to the behaviour of RC' described by the language {`Ticket!.Visa?`, `Ticket!.eCash?`}), nevertheless, for instance, the occurrence of the sequence{`Ticket!.Ticket?.eCash!`} in $RC_5 \otimes$ RC' (as well as in $RC_6 \otimes$ RC') yields a deadlock marking that is a marking where no transition is enabled. This is because of an error in role-component RC': an internal decision is made (either `Visa?` or `eCash?`), when sending the `Ticket`, and not communicated properly to the environment [1].

We are finally ready to give adequate definitions of more flexible behavioral compatibility relations for roles, which are based respectively on the weak and the strong usability. Then, strong compatibility always implies weak compatibility:

Definition 3.3 (Weak and Strong optimistic compatibility). Let RC_1 and RC_2 be two weakly (resp. strongly) usable roles.
RC_1 and RC_2 are Weakly (resp. Strongly) Optimistic Compatible, noted $RC_1 \approx_{WOC} RC_2$ (resp. $RC_1 \approx_{SOC} RC_2$), iff $RC_1 \otimes RC_2$ is weakly (resp. strongly) usable.

Example 4. The roles RC_3 and RC_4, shown in figure 2, are weakly (but not strongly) optimistic compatible that is $RC_3 \approx_{WC} RC_4$ holds since $RC_3 \otimes RC_4$ is weakly usable. Indeed, $RC_3 \otimes RC_4$ satisfies the completion option. Besides, the two roles RC_5 and RC_7 shown in figure 3 are strongly optimistic compatible that is $RC_5 \approx_{SOC} RC_7$ holds since $RC_5 \otimes RC_7$ is strongly usable. Indeed, we can check that $RC_5 \otimes RC_7$ terminates successfully.

4 Context-Based Behavioural Substitutability for Roles

Our main interest is to define optimistic behavioural subtyping relations (reflexive and transitive) capturing the principle of substitutability [9] and namely taking into account the context of use (environment). First, we define two subtyping relations between roles that are based upon the preservation of the (weakly or strongly) utilizing of the former role by any role of its environment. Then, we show the existing link between compatibility and substitutability concepts.

Definition 4.1 (Weak and Strong optimistic substitutability). Let $RC_1 = (Behav_1, Serv_1)$, $RC_2 = (Behav_2, Serv_2)$, be two roles, $Serv_i = (Req_i, Prov_i)$, i =1, 2, such that: $Prov_1 \subseteq Prov_2$, $Req_1 \subseteq Req_2$.
1. RC_2 is less equal to RC_1 w.r.t Weak Optimistic Substitutability, denoted $RC_2 \leq_{WOS} RC_1$, iff $\forall$ Env $\in$ ENV(RC_1), Env weakly utilizes $RC_1 \Rightarrow$ Env weakly utilizes RC_2.
2. RC_2 is less equal to RC_1 w.r.t Strong Optimistic Substitutability, denoted $RC_2 \leq_{SOS} RC_1$, iff $\forall$ Env $\in$ ENV(RC_1), Env strongly utilizes $RC_1 \Rightarrow$ Env strongly utilizes RC_2.

The Weak (resp. Strong) Optimistic Substitutability guarantees the transparency of changes of roles to their environment. Namely, the weak (resp. strong) optimistic compatibility between the former role and its environment should not be affected by these changes. In both weak and strong subtyping relations, the (super-) role component RC_1 can be substituted by a (sub-) role component RC_2 and any environment of the former role RC_1 will not be able to notice the difference since: (a) the sub-role has a larger set of required and provided services ($Req_1 \subseteq Req_2$ and $Prov_1 \subseteq Prov_2$) than the super-role, and (b) any environment that weakly (resp. strongly) utilizes the former role is also able to weakly (resp. strongly) utilize the new role.

Example 5. Let us consider the roles RC_3 and RC_6. $RC_3 \leq_{WOS} RC_6$ holds since the unique environment that strongly (and then weakly) utilizes RC_6 is the role RC_7, and $RC_7 \otimes RC_3$ satisfies the completion option. These two roles RC_3 and RC_6 are not related by the strong subtyping relation that is $RC_3 \leq_{SOS} RC_6$ does not hold, since $RC_3 \otimes RC_7$ does not terminate successfully. Last but not least, consider the roles RC_5 and RC_6; $RC_5 \leq_{SOS} RC_6$ holds since the role RC_7 (which is the unique environment) that strongly utilizes RC_6 also strongly utilizes RC_5. Indeed, one can check that $RC_5 \otimes RC_7$ terminates successfully.

Property 4.1 (Hierarchy of subtyping relations). Let $\Re C = \{RC_1, \ldots, RC_n\}$ be the set of role components in the system. The relations $\leq_H$, H $\in$ {WOS, SOS}, are preorder (reflexive and transitive) on $\Re C$ and form a hierarchy: $\leq_{SOS} \Rightarrow \leq_{WOS}$.

The following core theorem of this paper states two fundamental properties of roles compatibility and their substitutability. First, substitutability relations are compositional: in order to check if Env $\otimes RC_2 \leq_H$ Env $\otimes RC_1$, H $\in$ {WOS, SOS}, it suffices to check $RC_2 \leq_H RC_1$, since the latter check involves smaller roles and it is more efficient. Second, compatibility and substitutability are related as follows: we can always substitute a role RC_1 with a sub-role RC_2, provided that RC_1 and RC_2 are connected to the environment Env by the same provided services that is: Req $\cap$ $Prov_2 \subseteq$ Req $\cap$ $Prov_1$. This condition is due to the fact that if the environment utilizes services provided by RC_2 that are not provided by RC_1, then it would be possible that new incompatibilities arise in the processing of these provided services.

Theorem 4.1 (Compositionality and compatibility preservation). Let $RC_1 = (Behav_1, Serv_1)$, $RC_2 = (Behav_2, Serv_2)$ be two roles where $Serv_i = (Req_i, Prov_i)$, i = 1, 2. Let Env = (Behav, Serv) such that Req $\cap$ $Prov_2 \subseteq$ Req $\cap$ $Prov_1$.

1. Env $\approx_{WOC}$ RC_1 and $RC_2 \leq_{WOS} RC_1 \Rightarrow$ Env $\approx_{WOC}$ RC_2 and Env $\otimes$ $RC_2 \leq_{WOS}$ Env $\otimes$ RC_1.
2. Env $\approx_{SOC}$ RC_1 and $RC_2 \leq_{SOS} RC_1 \Rightarrow$ Env $\approx_{SOC}$ RC_2 and Env $\otimes$ $RC_2 \leq_{SOS}$ Env $\otimes$ RC_1.

5 Conclusion and Related Work

The aim of this paper is to present a new approach to the definition of context-based behavioral compatibility and substitutability for role components. We proposed two new and flexible compatibility relations together with two optimistic subtyping relations between role-components taking into account the non-deterministic, the composition mechanism of roles, as well as the property preservation such as the completion and the proper termination of roles. We furthermore investigated some properties such as the preservation of the compatibility by substitutability as well as the compositionality of the proposed substitutability relations.

Related work. In [8], the concept of usability is used for analyzing web service based business processes. The authors defined the notion of usability of workflow modules, and studied the soundness of a given web service, considering the actual environment it will be used in. Based on this formalism together with the notion of usability, the authors present compatibility and equivalence definitions of web services. This approach is close to ours, since the compatibility of two workflow modules is related to our strong optimistic compatibility of role-components. Further, in that work, the notion of equivalence of web services is defined upon the usability concept, but it is not linked to the compatibility. Whereas our work addressed the substitutability of role-components together with the preservation of compatibility by substitutability. In [2], authors define the notion of compatibility and consistency between components of the agents and role. There the agent is compatible with a role if the agent (sub) goals are subset of (sub) goals of the role. Conversely a role is compatible with the agent if the (sub) goals of the role are subset of the agent (sub) goals. This compatibility relation indicates that the agent is highly suitable to fulfill the role. Nevertheless such a match between the agent and roles is not always possible. Then, a weaker relation is introduced, called consistency, which indicates that the goals of the agent and the roles do not conflict. This approach is close to the compatibility relations that have been proposed in our previous work [6], since it is related explicitly to the property preservation, and then one of the limitations is that the obtained compatibility relations are very strong. Whereas in the approach presented in this paper, we have proposed more flexible behavioural compatibility relations by taking into account the environment in which the agent-roles are involved. Last but not least, in [4] the authors contextualize commitment protocols by adapting them, via different transformations, on context and the agent's preferences based on that context. A protocol is transformed by composing its specification with a transformer specification, and the contextualization is characterized operationally by relating the original and transformed protocols according to protocol substitutability relations. There, the semantics of protocol substitutability relations are based on state-similarity to compare states occurring in runs of protocols. Nevertheless, in practice, checking state-similarity between protocols is not quite easy, since the runs of protocols are needed for that

purpose. In contrast, our approach proposed substitutability relations for roles ensuring correctness guarantees w.r.t their usability that is the preservation of the utilizing of roles by their environment.

References

1. De Alfaro, L., Henzinger, T.A.: Interface automata. In: Proc. of ESEC/FSE. Software Engineering Notes, vol. 26, 5, ACM, New York (2001)
2. Dastani, M., Dignum, V., Dignum, F.: Role Assignment in Open Agent Societies. In: AAMAS'03, pp. 489–495. ACM, New York (2003)
3. Cabri, G., Leonardi, L., Zambonelli, F.: BRAIN: a Framework for Flexible Role-based Interactions in Multi-agent Systems. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 145–161. Springer, Heidelberg (2003)
4. Chopra, A.-K., Singh, M.P.: Contextualizing Commitment Protocols. In: AAMAS 2006, pp. 1345–1352. ACM, New York (2006)
5. Hameurlain, N., Sibertin-Blanc, C.: Specification of Role-based Interactions Components in MAS. In: Choren, R., Garcia, A., Lucena, C., Romanovsky, A. (eds.) Software Engineering for Multi-Agent Systems III. LNCS, vol. 3390, pp. 180–197. Springer, Heidelberg (2005)
6. Hameurlain, N.: Formalizing Compatibility and Substitutability of Role-based Interactions Components in Multi-agent Systems. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 153–162. Springer, Heidelberg (2005)
7. Hameurlain, N.: A Formal Framework for Component Protocols Behavioural Compatibility. In: APSEC 2006, pp. 87–94. IEEE Computer Society, Los Alamitos (2006)
8. Martens, A.: Analyzing Web Service Based Business. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 19–33. Springer, Heidelberg (2005)
9. Liskov, B.H., Wing, J.M.: A Behavioral Notion of Subtyping. ACM Trans. on Programming Languages and Systems 16(6) (November 1994)
10. Szyperski, C.: Component Software-Beyond Object-Oriented Programming. Addison-Wesley, Reading (2002)
11. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology. ACM TSEM 12(3), 317–370 (2003)

Governing Environments for Agent-Based Traffic Simulations

Michael Schumacher[1], Laurent Grangier[2], and Radu Jurca[2]

[1] University of Applied Sciences Western Switzerland
Institute of Business Information Systems
CH-3960 Sierre
[2] Ecole Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland

Abstract. Multiagent systems may be elegantly modeled and designed by enhancing the role of the environment in which agents evolve. In particular, the environment may have the role of a governing infrastructure that regulates with laws or norms the actions taken by the agents. The focus of modeling and design is thus shifted from a subjective view of agents towards a more objective view of the whole multiagent system. In this paper, we apply the idea of a governing environment to model and design a multi-agent system that micro-simulates the Swiss highway network. The goal of the simulation is to show how traffic jams and accordion phenomena may be handled with appropriate local regulations on speed limits. A natural modeling would give segments the capacity to regulate the speed based on observed local events. We developed the simulation platform from scratch in order to accommodate our design choices and a realistic complexity. This paper presents in details our modeling choices, and first experimental results.

1 Introduction

Agent-based micro-simulations are becoming a popular application area of multiagent systems (MAS), in areas such as social sciences, traffic management, biology, geography, or environmental sciences. Agent technology has opened a whole new methodology for studying real-world complex systems by simulating every individual through an autonomous agent. Individual behavior can thus be easily modeled, and the MAS captures the aggregated behavior of the collective. These agent-based *micro-simulations* help understanding better an emergent reality or allows trying virtually some settings that would be very costly to test in reality. Traffic management is a typical example. For instance, a micro-simulation may help to visualize the effect of constructing new roads on the overall traffic.

Some multiagent systems (and a fortiori agent-based micro simulations) may be elegantly modeled and designed by enhancing the role of the environment in which agents evolve. In particular, the environment may have the role of a governing infrastructure that regulates with laws or norms any action within the

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 163–172, 2007.

system. This has the strong advantage of a flexible modeling and design, where the focus is shifted from a subjective view of agents towards a more objective view of the whole multiagent system.

In this paper, we show first experiments on how we apply the governing environment to the modeling and design of a micro-simulation of the Swiss highway network. The goal of the simulation is to show how accordion phenomena and traffic jams may be handled with appropriate local regulations on the speed limit. For example, adaptive speed limitations my be implemented in order to maximize the throughput of the network.

A natural model gives segments the capacity to regulate the speed based on locally observed events. Therefore, regulating highway segments perfectly captures the design of a governing environment. Because of the complexity of the simulation and our choice in the above described modeling, we developed a simulation platform from scratch. This paper presents in details our modeling choices for the simulation platform. First experimental results of our implementation are also eluded. The adaptive distributed speed regulation will however be the subject of another paper, as it is still under development.

The paper is organized as follows. Section 2 introduces and explains the notion of governing environment. Sec. 3 explains our problematic of traffic simulation in Switzerland. After discussing our global modeling in Sec. 4 following the governing idea, we describe how we model the agent behaviors in Sec. 5. In section 6, we discuss experiments. Section 7 concludes the paper.

2 The Governing Environment

Most research in multiagent systems (MAS) has focused on the internal capacities of agents, and not on the medium in which they evolve. This vision is however changing towards enhancing the function of the environment in MAS (see for instance [10,1]). Actually, such a vision was already implicit in the early days of software agent research. This is shown by a definition of an autonomous agent as *a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future* [3]. This description stresses the importance of the environment as the living medium, the condition for an agent to live, or the first entity an agent interacts with. Thus an agent is part of the environment. But it remains autonomous, so that the environment may not "force" the agent's integrity. It is in this environment that an agent (autonomously) senses and acts. The acting of the agent on the environment directly influences its future sensing, because the environment is changed by the agent actions.

Even if the notion of environment was stressed as a main component of MAS, most approaches have viewed it as something being modelled in the "minds" of the agents, thus using a minimal and implicit environment that is not a first-order abstraction, but rather the sum of all data structures within agents that represent an environment. This is a typical *subjective view* of the multiagent system inherited from distributed artificial intelligence, which contrasts with

an *objective view* that deals with the system from an external point of view of the agents [7]. This objective point of view sees the environment as a central component for the modeling and design of MASs. Multiagent simulations belong to the type of systems that most explicitly model the environment.

An appealing way to exert the necessary level of control out of agents is the use of a *governing infrastructure* to structure and shape the space of actions within a MAS [5]. This governing perspective mainly allows managing agent interactions from an external point-of-view. This has the strong advantage that agents may be defined independently, and that some control is overtaken externally. In the area of virtual organizations, the Electronic Institutions (EI) approach [4] does this by defining so-called *governors* which are middle agents that mediate all (communicative) actions within a MAS [1]. This solution has, however, important disadvantages. Providing each agent with a governor puts a heavy computational burden on the infrastructure. But, more importantly, middle agents do not capture a natural modeling for the functionality they are expected to fulfill, i.e. mediation of communication. The governing or regulating responsibility should be transferred from specialized middle agents to the environment of a MAS, calling for the *environment as a governing infrastructure* [8]. This can be done with the idea of a *programmable coordination medium* [2], which essentially defines *reactions to events* happening in a shared dataspace. This schema has the strong advantage to allow the definition of laws that not only regulate agent interactions, but also any happening within an environment. Overall, we expect that viewing the environment as a governing infrastructure simplifies the design of multiagent systems. We will show this in the area of agent-based micro-simulation applied to traffic management.

3 Micro-Simulation of the Swiss Highway Network

We modeled, designed and implemented an agent-based micro-simulation that captures the ideas of governing environment. The application area is the simulation of the whole highway traffic in Switzerland of about 1700 km (see Fig. 1). The platform is bound to a geographical information system[2] that allows zooming from the global country view to the local view of each vehicle. We did not develop over an existing platform for agent-based simulation, because the complexity of the problem is much too big. Furthermore, it would be difficult to capture our modeling. We therefore built a new platform from scratch.

Our final goal is to study adaptive and decentralized speed limitation to have an optimal car throughput. Actually, there are some settings in which modern highways perform very poorly. First, the *accordion* is a transient mode in which cars accelerate to a given speed S, only to brake to almost a full stop immediately after reaching the speed S. Secondly, *traffic jams* usually occur in highway segments preceding a bottleneck (e.g. tunnels or accidents). Our goalof

[1] All actions that the EI approach accounts for are communicative by nature.
[2] http://www.geotools.org

the simulation is to show that adaptive distributed speed limitations on the highway segments preceding (and including) the one where problems might appear will drastically decrease the negative effects previously discussed. Therefore we want to investigate whether speeding restrictions can increase the efficiency of highways, and to determine automatic speeding restrictions that optimize highway utilization. As a methodology we decided to develop an agent-based micro-simulation to investigate the above hypothesis and to determine optimal speeding policies. A distributed speed regulation needs to split highways into segments with constant length so that on each segment one speed limitation can be imposed. Constraints between the speed limit on neighboring segments have to ensure that the vehicles do not have to break too abruptly.

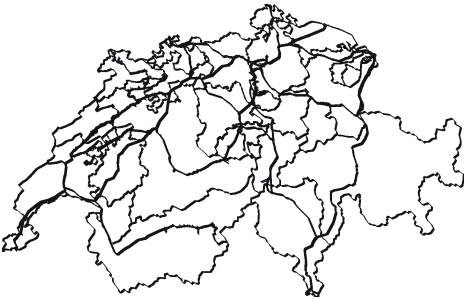


Fig. 1. Swiss national roads with limits of the cantons, as displayed in our simulation platform

An adequate modeling of a micro-simulation allowing distributed decision making on segments can elegantly use the paradigm of a governing environment. Actually, the segment naturally build the environment of the MAS. Each segment has a set of rules that regulate the state of the highway segment (number of cars, average speed, etc) and can decide on the speed limitation for that segment. Neighboring segments can propagate events to one another. Each vehicle is modeled by one agent which takes decisions based on a local view: a driver wants to get to the destination as fast as possible and guides her action depending on the traffic in her immediate vicinity. We further assume that drivers respect the speed limits (within certain bounds).

This paper reports the modeling of our simulation platform, and *not* the distributed adaptive decision process for optimal speed regulation. Actually, we are currently working on this with the DPOP [6] algorithm for distributed constraint satisfaction. This will be reported in a future paper.

4 Modeling

We describe in this section the modeling of the MAS of the micro-simulation. According to the *governing environment* paradigm, laws are defined within the environment. The environment reacts to raised events according to the rules that we define. Unlike the agents, the environment has no behavior and does not act itself: it can just react to events which are intercepted.

Static Model. We identified two types of agents that are organized around highway segments that represent the environment (see Fig. 2). The Vehicle class[3] has three state attributes : its position (relative to its current segment), its speed

[3] We use the UML profile described in [9], where rounded rectangles are agents.

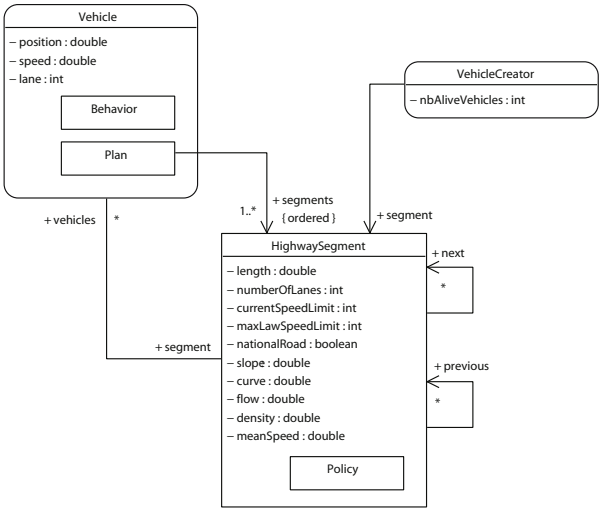


Fig. 2. Agent diagram of the system

and its lane position. Each vehicle has : a Plan which is an ordered collection of HighwaySegment telling it which way to take; a Behavior which describes its acceleration, deceleration and lane changing behavior. VehicleCreator is a dedicated agent which takes care of creating new agents in the system.

The highway is divided in segments. Each Vehicle lives in a HighwaySegment which can be considered as a continuous space. Each one is connected to its next following segments and its previous preceding segments. Vehicles can only move from their current segment to one of the next segments. HighwaySegment has a few constant attributes (length, numberOfLanes, maxLawSpeedLimit, nationalRoad, slope, curve) and a few variable attributes (currentSpeedLimit, flow, density, meanSpeed). All these attributes are part of the environment and can be perceived by agents.

Dynamic Description. The time of the simulation is discrete. We send a time step message to every agent at each step of the simulation, and they return an action depending on their perception and their internal behavior. The environment has a governor role and can react to some events.

The environment generates events. SpeedPolicyChangedEvent is generated by a segment each time the speed restriction is changed in a segment. The governing environment will tell the neighbor segments to reconsider their current speed limit. StepBeginEvent is an internal event which is generated by the environment itself to warn the segment that a time step has begun. StepEndEvent is the same type of event as StepBeginEvent, but it warns the segment against the end of a time step event. VehicleDestructedEvent is raised by the environment each time a vehicle finished its planning and should die. VehicleDensityChangedEvent is raised by a segment each time the density of the segment has changed. It tells the environment to reconsider its speed limit.

Agent actions generate events. VehicleCreatedEvent is launched by VehicleCreator each time it creates a new vehicle. VehicleChangedLaneEvent is posted by Vehicle every time it changes its lane position. VehicleChangedSegmentEvent is posted by Vehicle every time it leaves a segment and enters a new one.

5 Behavior Models

Vehicle behaviors are described by two different but connected models: i) the *car following model* describes how a car speeds up and brakes, and the ii) the *lane changing model* describes how a driver decides to change lane.

Car Following Model. Our model is inspired by the Intelligent-Driver Model (IDM) from Martin Treiber[4], which makes the vehicle accelerate to its speed objective (see Alg. 1). it does not have a constant acceleration. It decreases from the initial acceleration (a) to zero when approaching the speed objective (s_o). The deceleration value increases from b and is not limited in the theoretical model. Because of this, the vehicles can have unrealistic deceleration, but the system is collision free.

Algorithm 1. IDM car following model (acceleration computation)

Require: v, v_f, s, T, v_{limit}, a, b, s_{min}

1: $v_o \leftarrow humanizeSpeed(v_{limit})$
2: $\Delta_v \leftarrow v_f - v$
3: $s^* \leftarrow max\{s_{min}, s_{min} + vT + \frac{v\Delta_v}{2\sqrt{ab}}\}$
4: $a_c \leftarrow a\ 1 - \left(\frac{v}{v_o}\right)^4 + \left(\frac{s^*}{s}\right)^2$
5: **return** $max\{-3b, min\{a_c, a\}\}$

In Alg. 1, T is a the safety time with the ahead vehicle, values can be from 0.8 to 2 seconds. Here we use a normal distribution ($\mu = 1.5$, $\sigma = 0.5$) for this value. a is the maximum acceleration (0.8 m/s^2 for cars, 1.5 m/s^2 for trucks). b is the minimum deceleration (-2.5 m/s^2 for cars and trucks).

This model has interesting advantages since it is not based on the fact that vehicles will always keep a safe distance with the vehicle ahead. On the other hand, deceleration can be high and this can lead to bizarre behaviors, like when cars drive at a high speed and suddenly brake down with a high deceleration because of a traffic slow-down or a slowest car.

Lane Changing Model. Each vehicle must at each iteration consider changing lane or not. This decision is based on two main criterions for the agent : *is it safe to go on the other lane?* (safety criterion) and *do I get a reward to go on the other lane?* (incentive criterion). In our model, the safety criterion just says that the car behind would be able to brake in order to avoid a collision. We also check that the car ahead is not to close and that if it brakes, we will have the

[4] http://www.traffic-simulation.de

time to avoid a collision. The incentive criterion is quite simple. Vehicles change lane every time they can increase their speed on the other lane. Furthermore we add a few biases to make vehicles go to the right lane whenever the highway is going to change from a N lanes to N-1 lanes. Informally, this gives algorithm 2.

Algorithm 2. Basic lane changing model (incentive criterion)

```
1: if lane will end soon and already on the correct lane then
2:     ← do not change lane
3: end if
4: if lane will end soon AND not on the correct lane then
5:     ← go to right lane with an increasing probability when approaching to the end
       of the lane
6: end if
7: if already changed lane in the last 10 seconds then
8:     ← do not change lane
9: end if
10: if distance to car ahead is more than 200 meters then
11:     ← do not change lane
12: end if
13: ← change lane if we can increase speed on the other lane
```

The two first conditions make the vehicle go to the right lane if its lane will end soon. The third condition (line 8) avoids an oscillation movement from a lane to another. Imagine five vehicles on the right lane, and no vehicles on the left lane. They all have an incentive to go the left lane, once they changed, there is no vehicles on the right lane, so they all have an incentive to change for the right, and so on. They will all change at each step to the other lane. Condition at line 10 tries to avoid cars going to the left lane when they have no other car in front of them.

Combining the Car Following and the Lane Changing Models. Alg. 3 presents how to execute the car following and the lane changing models together. It ensures that all agents will have the same information when taking the decisions. A problem can occur when two vehicles compete for the same lane and think it is safe. They both will have an incentive to change for the target lane and both think that it is safe. To avoid it, we only change to the right at odd time steps and change to the left at even time steps.

Generation of Vehicles and plans. For the *generation of vehicles*, we used the number of registered vehicles in the canton (swiss regions) where the segment is. We put a defined percentage of vehicles (N) on highways. We also generate trucks on the basis of country statistics. Concerning the starting place, vehicles are created uniformly in the canton. Therefore every canton generates a predefined flow of vehicles in respect to its registered car population. Because official data from the Swiss Federal Roads Authority[5] were not of sufficient granularity,

[5] http://www.verkehrsdaten.ch/downloads/AVZ-StandorteStand012005.pdf

Algorithm 3. Vehicle state update loop

```
1: for all the vehicles do
2:    state ← current state of the environment
3:    decide to change lane or not according to state
4: end for
5: for all the vehicles do
6:    change lane if decided
7: end for
8: for all the vehicles do
9:    newState ← current state of the environment
10:   acceleration ← compute the new acceleration according to newState
11: end for
12: for all the vehicles do
13:   update the speed
14:   update the position
15: end for
```

we decided to create cars continuously. A realistic simulation should take into account different timing.

Each generated vehicle immediately has a deterministic assigned route plan, which can not change. This plan is however generated randomly. In future work, we will use demographic statistics and short-path algorithms to generate more realistic plans.

6 Experiments

Vehicle Generation. As said in section 5, we can calibrate the simulation to generate a percentage of the registered vehicles. This is difficult since knowing how many vehicles can drive simultaneously on Swiss highways is not obvious.

Swiss highways are composed of 1'855 km of roads. Since these roads have two possible directions and can have multiple lanes, the total length of lanes is about 7'550 km. Supposing a high congestion of 40 vehicles/km everywhere (this means one car each 25 meters on every lane and every highway segment), this leads to an estimation of 302'000 vehicles. It means that $N = 6\%$ of the Swiss vehicles would be on the highways. It can seem very low but, we should not forget that all the cars are never used at the same time and that there is a lot of other roads than highways in Switzerland. And of course, in reality at some place there is much more vehicles than at others, 40 vehicles/km is just an overestimated value of what could be a maximal congestion level.

We have made tests with different values of N (the maximal percentage of alive vehicles at a precise time). Table 1 shows how many vehicles can be simultaneously alive and how much time it costs to simulate a certain time. The first remark is about the theoretical value which is not equal to the practical one. It comes from the way of generating vehicles. Each creator segment has a physical maximum flow of vehicles and depending of the local conditions (i.e. a traffic jam on this segment), it can be lower that what it should be to ensure

Table 1. Maximum number of vehicles with respect to N

N	Theoretical	Practical	Simulated time [h]	Real time [h][6]
10 %	492'230	249'000	1:30	24:00
5 %	246'115	192'000	2:35	60:00
2 %	98'445	95'000	1:00	8:00

Table 2. Mean flow measurements with respect to N

Place	Real flow	$N = 10\%$	$N = 5\%$
Muttenz	10700	4140	4551
Wuennewil	2342	3318	3533
Grandvaux	5662	3941	3798
Monte Ceneri	3243	3432	3732
Giessbachtunnel	983	2053	2103
Erstfeld	2192	2143	1366
Bardonnex	3656	1834	1783
Oftringen	5928	3304	3625

the theoretical production of cars. Thus it is absolutely normal to have a lower value.

Tests of the Models. We ran the simulation with different values of N and looked at some randomly chosen place to see if the flow of vehicles we simulate is near reality or not. Vehicles were not always perceiving the current reality and were basing their decision on a partial future state. This was leading to many collisions, but since they are automatically cleared[7], the simulation was realistic. Table 2 shows the measures we found depending on the N value. The simulated time is the value given in table 1.

Values are very far from reality. However we remark that where there is a high mean flow value in reality, there is also a relative high mean flow in the simulation. This lets us think that even if our vehicle generation method is not realistic, it does not give arbitrary values.

7 Conclusion

We developed a micro-simulation of the Swiss highway network in order to show that the governing environment can be useful for MAS development. In our simulation platform, the design has shown to be very flexible. Future work will consist in improving the vehicle behavior modeling and the performance of the platform, and in actually implementing the adaptive and distributed speed limit regulations in order to achieve an optimal car throughput. We shortly explain hereafter those points.

[6] Tests made on a 4 x 3 Ghz 64-bits processors computer with 4 Gb RAM.

[7] The vehicle which causes the collision (the vehicle at the back) is deleted and everything continues as if nothing happened.

The model of a vehicle should become more realistic. Collisions should be avoided when two segments merge in one, including highway entries. We think this is very tricky to solve since road granularity information is not detailed enough to let us have finer grained models. The lane changing model should also be improved, especially at the end of lanes (when N ways merge to N-1 ways). Our model is not yet very good and produces unrealistic traffic jams.

Running a simulation with hundreds of thousands of agents is not costless. To simulate a real scenario with many vehicles in a reasonable time, we have to make deeper changes in the architecture. A way to do it is to distribute the computation on several computers. We estimate that our architecture should be easily transformable into a distributed one, for instance with segment distribution and asynchronous events.

However, the most important remains the realization and testing of an intelligent distributed speed restriction policy. We are currently working on this using a distributed constraint optimization algorithm called DPOP [6].

References

1. Weyns, D., Van Dyke Parunak, H., Michel, F., Holvoet, T., Ferber, J.: Environments for Multiagent Systems, State-of-the-art and Research Challenges. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, Springer, Heidelberg (2005)
2. Denti, E., Natali, A., Omicini, A.: Programmable Coordination Media. In: Garlan, D., Le Métayer, D. (eds.) COORDINATION 1997. LNCS, vol. 1282, pp. 274–288. Springer, Heidelberg (1997)
3. Franklin, S., Graesser, A.: Is it an Agent or just a Program? A Taxonomy for Autonomous Agents. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) Intelligent Agents III. Agent Theories, Architectures, and Languages. LNCS, vol. 1193, pp. 21–35. Springer, Heidelberg (1997)
4. Noriega, P., Sierra, C.: Electronic institutions: Future trends and challenges. In: Klusch, M., Ossowski, S., Shehory, O.M. (eds.) CIA 2002. LNCS (LNAI), vol. 2446, pp. 18–20. Springer, Heidelberg (2002)
5. Omicini, A., Ossowski, S., Ricci, A.: Coordination infrastructures in the engineering of multiagent systems. In: Bergenti, F., Gleizes, M.-P., Zambonelli, F. (eds.) Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, June 2004, ch. 14, pp. 273–296. Kluwer Academic Publishers, Dordrecht (2004)
6. Petcu, A., Faltings, B.: Dpop: A scalable method for multiagent constraint optimization. In: IJCAI 05, Edinburgh, Scotland, pp. 266–271 (August 2005)
7. Schumacher, M.: Objective Coordination in Multi-Agent System Engineering. LNCS (LNAI), vol. 2039. Springer, Heidelberg (2001)
8. Schumacher, M., Ossowski, S.: The governing environment. In: Weyns, D., Parunak, H.V.D., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 88–104. Springer, Heidelberg (2006)
9. A UML Profile for Agent-Oriented Modeling: In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) AOSE 2002. LNCS, vol. 2585, Springer, Heidelberg (2003)
10. Weyns, D., Schumacher, M., Ricci, A., Viroli, M., Holvoet, T.: Environments in multiagent systems. Knowledge Engineering Revue 20(2), 127–141 (2005)

Knowledge Driven Architecture for Home Care

Ákos Hajnal[1], David Isern[2], Antonio Moreno[2],
Gianfranco Pedone[1], and László Zsolt Varga[1]

[1] Computer and Automation Research Institute of the Hungarian Academy of Sciences,
Kende u. 13-17,
1111 Budapest, Hungary
{ahajnal,gianfranco.pedone,laszlo.varga}@sztaki.hu
[2] ITAKA research group - Intelligent Technologies for Advanced Knowledge Acquisition
Computer Science and Mathematics Department, University Rovira i Virgili
Av.Països Catalans, 26. 43007-Tarragona, Spain
{david.isern,antonio.moreno}@urv.cat

Abstract. Multi-Agent Systems (MAS) in health-care domains are showing a rapid increase, in order to manage complex tasks and adapt gracefully to unexpected events. On the other hand, the lack of well-established agent-oriented engineering methodologies to transform knowledge level descriptions into deployable agent systems slackens MAS development. This paper presents a new methodology in modelling and automatically implementing agents in a home care domain. The representation of the application knowledge together with the codification of health care treatments lead to flexible realization of an agent platform that has the capability to capture new medical knowledge emerging from physicians.

Keywords: knowledge driven architecture, home care, agent-based care system, model-driven agent generation, emerging knowledge management.

1 Introduction

Engineering software systems needs powerful abstractions to handle complexity. The agent paradigm advances the modelling of software systems by embodying a stronger notion of autonomy and control than objects, including the notion of reactive, proactive and social behaviours, as well as assuming inherent multi-threaded control. In order to be able to build complex and reliable systems, we need not only new models and technologies, but also an appropriate set of software engineering abstractions that can be used as a reference for system analysis, and as the basis of the methodologies that enable developers to engineer software systems in a robust, reliable, and repeatable fashion. A well accepted approach to engineer object oriented software systems is the Model Driven Architecture (MDA) [1] which provides a new way of writing specifications, based on a high level platform-independent model. The complete MDA specification consists of a platform-independent base UML model, plus platform-specific models and interface definition sets, each describing how the base model is implemented on different target platforms. There are several agent-oriented software engineering methods, however they do not contain the transformations

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 173 – 182, 2007.

from high level models into platform-specific models in the way as it is in the MDA approach. In this paper we are advancing agent oriented software engineering methods by applying an MDA style approach to the engineering of the agent platform of the *K4Care* project in order to support home care. We call this software engineering approach as *knowledge driven architecture for home care*. There are two main differences from MDA: one is that the high level description is knowledge based compared to the object oriented model of MDA; the other is that the target is a platform concerning the agent paradigm. These advancements are perceived as a considerable support when implementing software for a knowledge intensive application.

The aim of the K4Care European project is to provide a *Home Care model*, as well as design and develop a prototype system, based on Web technology and intelligent agents that provide the services defined in the model. The percentage of old and chronically ill people in all European countries has been growing steadily in the last years, putting a very heavy economic and social pressure on all national Health Care systems. It is widely accepted that this problem can be somehow palliated if *Home Care* (HC) services are improved and used as a valid alternative to hospitalisation.

In the context of the K4Care project, we need a software engineering method that (at least partially) automates the development of an agent platform for a knowledge intensive application like home care. The intended application has two basic features: a) actors are members of well defined organizations, b) there is extensive domain knowledge to be considered. Despite similarities with the MDA approach, we have to take into account that MDA is based on UML, which is basically a notation language, not a specification one. Nevertheless, UML does not capture the behavioural aspects (dynamism) of run-time interactions. The main innovations of the paper's work are the following:

- A general methodology proposal in modelling an organizational domain, capturing and classifying the application knowledge, categorized into two main classes: *declarative* (identifying the system compositional elements) and *procedural* (capturing the behavioural logic governing the system).
- The capability of the system to capture the knowledge emerging from the behavioural skills of agents orchestrated by the interaction artefacts (SDA* Language [2]). Procedural knowledge is codified through elementary entities (States – Decisions – Actions) that, combined together, can capture the knowledge manifested at run-time by the physicians.
- Automation in the software engineering of the agent-oriented code.

The result of our investigations is a knowledge driven software engineering architecture that supports agent code generation from ontologies and codified documents for treatments, and follows a specific agent-oriented software engineering methodology.

The rest of the paper is structured as follows. Section 2 comments the most well-known paradigms in agent-oriented software engineering and the contribution of this work together with agent code generation from knowledge descriptions. Section 3 describes the *knowledge-driven architecture*. Section 4 reports the state of the realization of the proposed knowledge driven architecture in the K4Care project. The paper finishes with a brief conclusion.

2 Software Technology for Agent Systems

In the actual state of the art there is still a lack of well-established Agent-Oriented Software Engineering (AOSE) methodologies to support the automatic transformation of system description into deployable agent system. Several AOSE methodologies [3] have been proposed. Some of them offer specification-based formal verification, allowing software developers to detect errors at the beginning of the development process (e.g. Formal Tropos [4], [5]). Others inherit Object-Oriented (OO) testing techniques to be exploited later in the development process, upon a mapping of agent-oriented abstractions into OO constructs (e.g. PASSI [6], INGENIAS [7]). In addition, a structured testing process for AOSE methodologies that complements formal verification is still missing. From a theoretical point of view, different metaphors have also been proposed in the literature. The ant algorithms metaphor ([8], [9]) has shown to be useful in efficiently solving complex distributed problems. The physical metaphors ([10]) focus on the spontaneous reshaping of a system's structure and may have useful applications in pervasive and mobile computing. The societal metaphors have been effectively applied in robotics applications ([11]) and in the understanding and control of highly decentralized systems ([12]).

A formal AOSE modelling technique has been provided by the GAIA methodology ([13]), whose approach focuses on the development of large size systems that have to guarantee predictable and reliable behaviours. The metaphor adopted by GAIA is the *human organization*, which may be the most appropriate for highly populated domains ([14], [15]). A computational paradigm must be enabled by the proper technology: Multi-Agent System (MAS) platforms are the necessary infrastructures to support agents and their behavioural model. In the K4Care project we use JADE ([16]), which is a FIPA ([17]) compliant middleware for the development and run-time execution of agent-based applications. Mutually accepted standards in MAS conceptualization and implementation have not reached an adequate level of diffusion. In this context of methodological uncertainty ontologies (expressed in terms of OWL) may play an important role at modelling time, targeting the possibility to automate the engineering of the agent software. In this paper we propose the development of a complete Agent-based System for the home care domain, taking into account knowledge coming from different ontological sources (the domain model, the project requirements, the implementation and technological issues) and from medical procedures representation. Instead of separately using an AOSE methodology to model the MAS and consequently developing the inherent code, we merged the two aspects by conceptualizing all inherent elements into ontologies, interpretable descriptions, and then generating the deployable code. The interpretable descriptions contain the following knowledge: the *domain*, the *agent-paradigm*, the *standards* (GAIA methodology, FIPA) and the *technology* enabling the agent paradigm (JADE). It is important to highlight the complexity and the innovation of the paper's work. Others in the literature ([18], [19]) have demonstrated that the problem of deriving code from an ontology is not a trivial task due to different factors: the *higher semantics* of the OWL language in comparison with programming languages (like JAVA, in this project); the *complexity of the domain* representation, the relations and the knowledge contained, and the *run-time code dependencies* that must be captured and included in the automation.

3 Knowledge Driven Architecture for Home Care

Developing software for medical applications and in particular for home care is special in the sense that the developed software needs to incorporate and support complex and sometimes intuitive knowledge present in the head of the medical staff, and also the medical staff is organized into a well structured organisation where the roles and responsibilities of each participant are well defined.

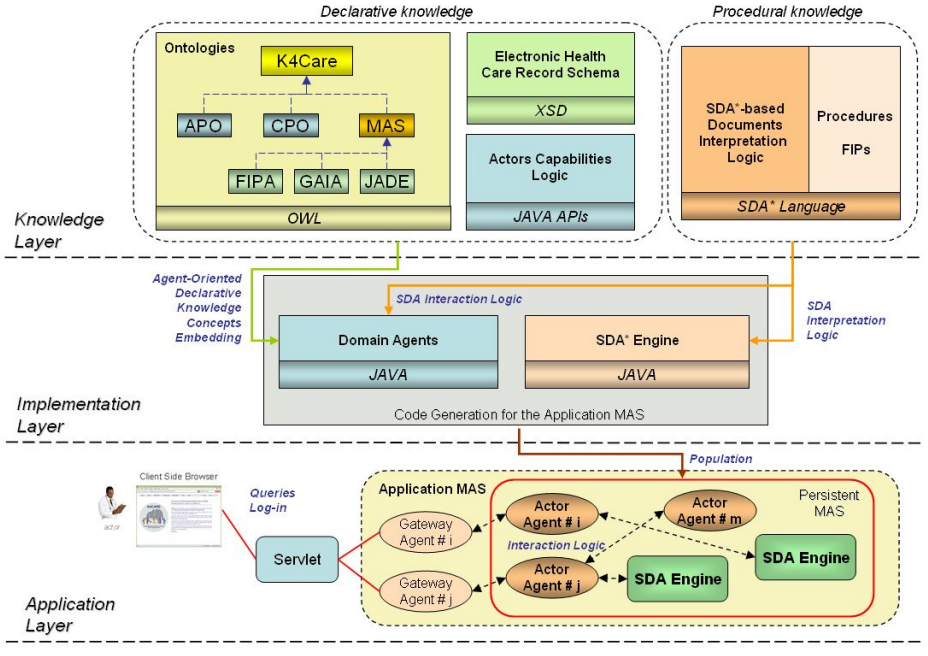


Fig. 1. Knowledge-Driven Architecture for Home Care

This reason led us to implement the software system in the K4Care project by starting from a knowledge based domain description, then deriving the software design using the GAIA methodology to achieve an agent implementation on a JADE agent platform. The most relevant aspect of this knowledge driven architecture is the separation of knowledge description from the software realization, granting a high level of interoperability and, at the same time, high level of independence among elements of the system. Key elements of the architecture are the *declarative knowledge*, the *procedural knowledge*, the *agent-oriented code generation*, the *agent behavioural logic distribution*, and the *interaction between agents and end-users*. The knowledge driven architecture for home care and its elements are shown in Figure 1.

The declarative knowledge and the procedural knowledge form the Knowledge Layer. The agent code generation forms the Implementation Layer. Finally, the agent behavioural logic distribution and the interaction between agents and end-users form

the Application Layer. The Knowledge Layer describes user's knowledge in a high level and the Application Layer is the targeted K4Care agent platform. In the following we are describing these elements.

3.1 Declarative Knowledge

Ontologies are a standard AI knowledge representation mechanism ([20]). Their main components are a set of concepts (or classes) *C*, which are taxonomically related by the transitive *is-a* relation and non-taxonomically related by named object relations $R^* \varepsilon C^*C^*String$. They represent the knowledge assets of the K4Care application and the catalyst for the agent's behavioural model, as well as the fundaments for the agent's code generation. We have divided the *application ontologies* into different sources, in relation to their application coherence. All the ontologies are independent of each other, except of the *K4Care* global one, which includes all the others and contains necessary cross-references to relate concepts coming from different elementary ontologies. The application ontologies are:

- **Domain ontology**, divided into *Actor Profile Ontology* (APO) and *patient-Case Profile Ontology* (CPO). APO represents the profiles of the subjects in the *K4Care* model (healthcare professionals, patients and social organisms) and contains the skills, concerns, aspirations of the people that they represent, together with the healthcare services that those people offer to or receive from the *K4Care* model. CPO represents the relevant medical concepts (e.g. symptoms, diseases, syndromes). Domain ontologies describe "know-what" knowledge about actors accessing the *K4Care* model and pathologies the *K4Care* model gives support to.
- **MAS ontology**, whose double role is to establish a common conceptualization of the Multi-agent System architecture and to enable the automation of its code generation. The MAS ontology includes the following elements:
  - **FIPA ontology.** This is the general conceptualization in the standardization of a MAS development. It represents "know-what" knowledge of a MAS (messages, encodings, communication languages, and so on) and "know-how" knowledge of their interactions (communicative acts, interaction protocols, communication ontologies, and so forth).
  - **GAIA ontology.** It includes the ontological description of the MAS development methodology adopted within the K4Care project, with particular attention to concepts such as *role*, *responsibility*, *permission* ([13]).
  - **JADE ontology.** It expresses the implementation and deployment concepts that must comply with the elected MAS of the project, i.e. JADE agent-types, behaviours, containers, mobility, and so forth ([16]).
- ***K4Care* ontology.** This ontology is necessary in order to model the inevitable ontology-cross references: it represents the way implementation ontologies (MAS) previously mentioned are connected to the domain specific ontologies (APO, CPO). The agent capabilities, for example, are expressed in terms of "*actions*" in the APO; these are considered as "*responsibilities*" or

"*permissions*" by GAIA; the behavioural logic of an agent inherent to its capabilities is expressed in JADE in terms of "*behaviours*", which can contain FIPA compliant "*interaction protocols*", as well. Despite its undoubted importance, this ontology has a small dimension, in terms of relations and items contained.

The *Electronic Health Care Record* in Figure 1 represents the patient's care profile and it is consulted by physicians during the stages of a treatment. *Actor Capabilities*, in addition, are expressed in terms of primitives and they are developed to define an actor's basic skill within the system. They represent the agent's permanent capabilities. The global upper-level ontology (composed in OWL) containing all the relevant concepts in K4Care project consists of about 500 classes, and 70 properties.

3.2 Procedural Knowledge

In the K4Care project we have separated the "*static agent capabilities*", that are defined through the APO ontology in terms of "*actions*", and represents "*what*" an agent is able to do, from the "*dynamic behavioural capabilities*", that emphasize the way static agent capabilities can be combined together in order to define much more complex emerging *interactions* following the care service procedures of the domain. The procedural knowledge is formally described in different fundamental elements:

- *Procedures*: descriptions of the steps to be taken within the *K4Care platform* to provide one of the HC services.
- *Formal Intervention Plans (FIPs)*: general descriptions, defined by health care organisations, of the way in which a particular pathology should be treated.
- *Individual Intervention Plans (IIPs)*: descriptions of the specific treatment that has to be provided to a particular patient.

Procedural knowledge codifies complex medical tasks and is required to define the set of available actions performed by all actors in the platform. This knowledge is expressed in the K4Care project using a flowchart-based representation called SDA* ([2]). A small example of a formal SDA*-based representation (depression with dementia) is shown in Figure 2.

3.3 Agent Behavioural Logic Distribution

A single actor (agent) in the platform owns a limited set of behavioural capabilities that represents its role in the home care organizational context. Despite this limited number of skills, the agents' capabilities can be orchestrated and combined by physicians in order to obtain much more complex global behaviours from the entire system. New treatments are coded into SDA* graphs that are interpreted at run-time by a special agent (SDA* Engine in Figure 1), whose objective is to govern the complexity of the treatments representation and to lead agents interactions (distribution) toward the provision of the requested services. This gives the system a powerful and elegant flexibility in managing and executing new treatments and medical guidelines.

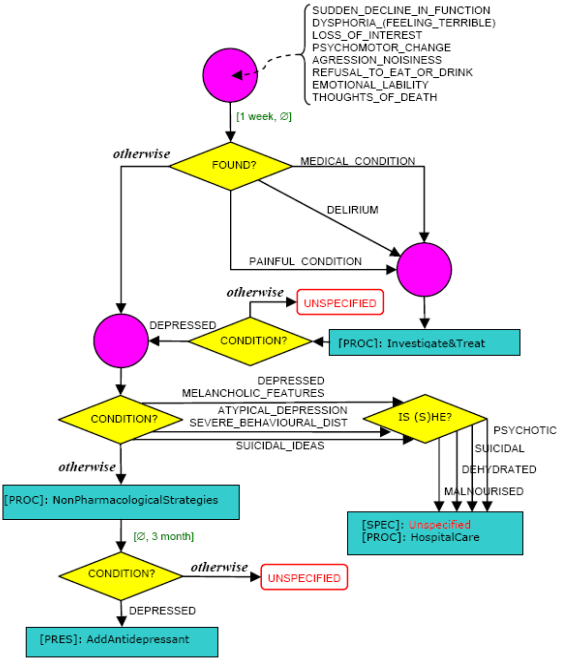


Fig. 2. SDA* to manage the Depression with Dementia

The interaction level of an SDA* graph is determined by evaluating the treatment parameters that characterize a specific node in the graph (patient conditions) first, and then accessing the correct interaction relation. These relations can be imagined as a space of 4-tuples, whose components indicate a *subject* (the actor who is asked to provide an action in the specific node), an *object* (the actor who will receive the benefits of the *subject*'s action), a *service* (the type of service requested) and a *document* (this represents the document that must be officially updated during the patient's treatment, in compliance with a specific procedure).

3.4 Interaction Between Agents and End-Users

There is a clear separation between the view, the control and the model within the K4Care project. The user interface (*view*) is connected to the MAS (*model*) via a servlet (*controller*) and all communications towards and from the multi-agent system occur by the support of a gateway-agent (one per permanent agent/actor in the MAS), which is strictly coupled with the servlet. The front-end of the interaction is represented by a web-application, whose structure and content are customized in relation with different relevant factors (such as the patient profile's insight or the interaction device – PC, PDA, Notebook).

4 Knowledge Driven Architecture Realization in K4Care

We applied the proposed knowledge driven architecture to the development of the K4Care system. We have realized the knowledge-based description of the domain knowledge (ontologies, procedural documents, agents capabilities description). Our experience demonstrated that medical experts find it sufficiently easy to describe and validate their expertise using ontologies. Despite the undoubted but minimal initial effort in familiarizing with ontology theory and supporting tools, medical experts considered our approach an elegant and powerful method for homogeneously representing medical knowledge (together with treatment guidelines documents). We have already realized all domain ontologies, defined within K4Care: the APO, the CPO and the MAS ontology (FIPA+GAIA+JADE) introduced in the previous section. K4Care global ontology contains the object properties that connect and bridge the different ontological layers (as illustrated in Figure 1).

As already mentioned, an important aspect of our project is the automation in the generation of agent code. Starting from the application knowledge, the accuracy and the level of detail in the code is directly proportional to the formalism's accuracy in describing the domain model. Particularly important is the representation of the agent capabilities, which, in practice, consists of a formal description of their business logic through API invocations. The objective of this automation process was to analyze, recognise and extract all the MAS compliant agent-oriented information: we had to take into account, first of all, the implementation requirements deriving from the elected MAS structure (agent class definition, protocols, message structures, message contents, behaviours class). In order to guarantee flexibility in the ontological traversing and parsing, and to decouple the conceptualization layer from the parsing one, the introduction of a mapping *dictionary* was necessary. This *thesaurus* described to the parser the "root" level of concepts where to start collecting agents' knowledge from. All relevant extracted information has been temporarily represented in an internal model. We have obtained agents code by combining together two categories of elements: *fixed and invariable* (keywords related to programming language, JAVA, and the MAS, JADE), and *knowledge-model-derived*, necessary to embed expressions which comply with the correct code completion of an agent. The agent's capabilities (domain knowledge) are represented by its actions, listed in the APO: from the paradigm's point of view, these must be translated in terms of behavioural models accepted by the MAS. The code was built by taking into account the following aspect: different behaviours have been dedicated to different skills in order to interact in parallel with other entities. Most of the effort in the agent code generation was also dedicated to obtain a harmonious code arrangement of all concepts (elements) characterizing the domain model and the application development.

The actual state in the realization of the described methodology highlights its real feasibility. Application ontologies are formalized in sufficient detail with respect to the goals of the project. Further details of the automated agent code generation based on ontologies are omitted here for space considerations.

5 Conclusions

We have presented a novel knowledge driven architecture for agent software development. The main feature of our architecture is represented by the fact that the development starts from the description of the knowledge involved in the home care domain (divided into declarative/static and procedural/dynamic/emerging) and then automatically derives the agent system implementation for the K4Care agent platform. Among others, a fundamental aspect of our approach is the platform's capability to recognize, manage and embed the emerging of new medical knowledge, expressed in terms of new SDA*-based patient treatments. This aspect extends the actor's (agent) capabilities, by deriving and combining their starting behavioural model, without the necessity to re-plan or re-implement the code structure of the agents. We have defined the knowledge driven architecture and implemented the knowledge layer with the help of medical staff. According to the experiences the knowledge layer is highly flexible and user friendly. This paper described the general concept of the novel knowledge driven architecture, leaving out the details like the agent code generation or the SDA* Engine due to space limitations.

Acknowledgements

The authors would like to acknowledge the work of all the K4Care partners, especially Fabio Campana, Roberta Annicchiarico and the medical partners (K4Care model), David Riaño (SDA* formalism), Sara Ercolani, Aïda Valls, Karina Gibert, Joan Casals, Albert Solé, José Miguel Millán, Montserrat Batet and Francis Real (ontologies, data abstraction layer and service execution). This work has been funded by the European IST project *K4Care: Knowledge Based Home Care eServices for an Ageing Europe* (IST-2004-026968).

References

1. OMG Architecture Board ORMSC: Model driven architecture (MDA). OMG document number ormsc/2001-07-01 (July 2001), http://www.omg.org
2. Riaño, D.: The SDA model v1.0: A set theory approach. Technical Report (DEIM-RT-07-001), University Rovira i Virgili (2007), http://deim.urv.es/recerca/reports/DEIM-RT-07-001.html
3. Henderson-Sellers, B., Giorgini, P. (eds.): Agent-oriented methodologies. Idea Group Publishing, USA (2005)
4. Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., Traverso, P.: Specifying and analyzing early requirements in Tropos. Requirements Engineering 9(2), 132–150 (2004)
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20(1-2), 3–50 (1993)
6. Cossentino, M.: From requirements to code with the PASSI methodology. In: [3], ch. IV, pp. 79–106 (2005)
7. Pavon, J., Gomez-Sanz, J., Fuentes, R.: The INGENIAS methodology and tools. In: [3], ch. IX, pp. 236–276 (2005)
8. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: From natural to artificial systems. Oxford University Press, New York (1999)

9. Babaoglu, O., Meling, H., Montresor, A.: Anthill: A framework for the development of agent-based peer-to-peer systems. In: Proceedings of the 22nd International Conference on Distributed Computing (2002)
10. Mamei, M., Zambonelli, F., Leonardi, L.: Distributed motion coordination with co-fields: A case study in urban traffic management. In: Proceedings of the 6th Symposium on Autonomous Decentralized Systems, p. 63. IEEE Computer Society Press, Los Alamitos (2003)
11. Moses, Y., Tennenholtz, M.: Artificial social systems. Computers and Artificial Intelligence 14(6), 533–562 (1995)
12. Hattori, F., Ohguro, T., Yokoo, M., Matsubara, S., Yoshida, S.: Socialware: Multiagent systems for supporting network communities. Communications of the ACM 42(3), 55–61 (1999)
13. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: the Gaia Methodology. ACM Trans. on Software Engineering and Methodology 12(3), 317–370 (2003)
14. Demazeau, Y., Rocha Costa, A.C.: Populations and organizations in open multi-agent systems. In: Proceedings of the 1st National Symposium on Parallel and Distributed Artificial Intelligence (1996)
15. Zambonelli, F., Jennings, N.R., Omicini, A., Wooldridge, M.: Agent-oriented software engineering for internet applications. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 326–346. Springer, Heidelberg (2002)
16. Java Agent DEvelopment Framework, http://jade.tilab.com/
17. Foundation for Intelligent Physical Agents, http://www.fipa.org/
18. Eberhart, A.: Automatic generation of Java/SQL based Inference engines from RDF Schema and RuleML. In: Proceedings of the First International Semantic Web Conference, pp. 102–116 (2002)
19. Kalyanpur, A., Pastor, D., Battle, S., Padget, J.: Automatic mapping of OWL ontologies into Java. In: Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering, pp. 98–103 (2004)
20. Fensel, D.: Ontologies: A silver bullet for knowledge management and electronic commerce. Springer, Berlin (2001)

MASL: A Logic for the Specification of Multiagent Real-Time Systems

Dmitry Bugaychenko and Igor Soloviev

Department of Computer Science
Saint-Petersburg State University
DmitryBugaychenko@tepkom.ru
soloviev@is1483.spb.edu

Abstract. In this paper we present a logic to provide a framework for the formal specification of multiagent real-time systems which allows explicit reasoning about the actions of agents, the nondeterministic model of interaction between agents and environment, the cooperation and competition of agents and the reaction time limits of a system. The logic combines Propositional Dynamic Logic PDL and Alternating-time Temporal Logic ATL and extends the formalism with reaction time constraints. We introduce a multiagent system abstract model and show how the logic can be used to specify the model properties.

Introduction

Most of modern complex and distributed intelligent systems are designed as a set of high-level autonomous interacting entities — *agents*. Such systems form a wide class of *multiagent* systems. What is believed to be the main reason of the intensive use of multiagent systems and a large number of their implementations is that the *intelligent agent* concept have such properties as: autonomous rational behavior, interaction with nondeterministic environment, reactivity, proactiveness and social ability. For an introduction to the intelligent agent concept refer to [1] and for information on multiagent systems applications — to [2] and [3].

Real-time control systems are one application domain of the multiagent systems. Reliability and adequacy are critical for such systems. Failures of such systems could cause damage to expensive equipment or even lead to human deaths. A variety of formal approaches including formal verification and automated testing are widely used to improve the reliability of such systems. These methods require a powerful formal mechanism of system specification which is often based on a variant of a *temporal logic*. For a review of different temporal logics see [4].

Different tools, languages and frameworks based on a variant of a temporal or predicate logic were used for the formal specification and modeling of multiagent systems. For example, DESIRE [5] framework is used to model multiagent system in [6], in [7] MetateM system is extended to Concurrent MetateM, which supports modeling of distributed systems, and the Z language [8] is used in [9] to specify

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 183–192, 2007.

properties of agents in multiagent systems. However, the classical temporal logic has no explicit means of specifying the cooperation and competition of agents and nondeterministic environment behavior, which hampers the usage of this logic for the multiagent systems specification.

This drawback was recently eliminated in Pauli's Coalition Logic CL [10] and Alur's Alternating-time Temporal Logic ATL [11], which allow explicit reasoning about the ability of agents and groups of agents (*coalitions*) to achieve certain goals in a nondeterministic environment with competing agents and groups of agents. These works immediately drew attention of the scientific community and were further developed. An epistemic extension of ATL was proposed in [12,13] and the concept of *role* was refined in the ATL formalism in [14]. Coalition logic CL was further investigated in [15] and [16].

Although these new approaches allow explicit reasoning of the cooperation and competition of agents, most of them still have no explicit means of reasoning about interaction with nondeterministic environment. This problem is dealt with in [17], where the authors propose using a combination of Propositional Dynamic Logic PDL [18] and Coalitional Logic CL to specify the ability of agents and coalitions to influence environment state transitions by performing joint actions. As far as we are concerned, this approach, which is herein referred to as $PDL + CL$, is the best choice for the foundation of multiagent real-time system specification logic.

By this paper we propose a logic $MASL$ that eliminates several shortcomings of $PDL + CL$, namely:

- Inability to specify *behavior* of a system, since $PDL + CL$ is intended for the specification of the system *ability*.
- Absence of the notion of *perception* and other means of reasoning about incomplete information.
- Inability to specify complex temporal properties of a system.
- Absence of means of specifying the reaction time limits of a system.
- Syntax mixture of two logics.

From the point of view of the multiagent system specification task, the main shortcoming of $PDL+CL$ is that it specifies what a system *can do*, but not what the system actually *does*. This shortcoming is eliminated in $MASL$ by using the refined and extended multiagent system model proposed in [19,1]. Since this model includes a notion of *perception*, we use it to reason about incomplete information too.

$PDL + CL$ temporal modalities are limited to the only "next time" modality, which restricts the expressive power of the logic. To avoid this shortcoming $MASL$ uses ATL instead of CL to specify temporal system properties. Thus, $MASL$ includes all the ATL temporal modalities: "always", "eventually", "until" and "next time".

No explicit means of specifying reaction time limits are included in $PDL + CL$, which is a significant shortcoming for the real-time systems specification. This shortcoming is eliminated in $MASL$ by using subscript index syntactical

construct and proper semantic extensions by analogy with $TCTL_s$ extension of the classical branching-time temporal logic CTL.

In $PDL + CL$, the specification of environment behavior is mixed with the specification of system abilities, which makes it difficult to use existing tools for PDL and CL. This shortcoming is resolved in $MASL$ by including environment specification in the *model* for formulas interpretation together with the multiagent system model.

The remainder of the paper is structured as follows. We begin with defining multiagent system model and $MASL$ logical foundation in Section 1. Then, in Section 2, we introduce the logic for the specification of environment behavior and model-checking algorithm for this logic. In Section 3, we introduce the logic for the specification of multiagent real-time systems. Finally, we conclude with some comments and a brief discussion of future work.

1 *MASL* Foundation

As a foundation for the multiagent system formal model, we use the intelligent agent model from [19] and extend it to deal with the case of multiple agents. A multiagent system is a pair $MAS = \{AG, S\}$, where $AG = \{ag_1, \ldots, ag_n\}$ is a finite non-empty set of agents and S is a finite non-empty set of environment states.

A system agent $ag \in AG$ is a tuple $ag = \{ACA_{ag}, I_{ag}, act_{ag}, see_{ag}, refn_{ag}\}$ where:

- ACA_{ag} is a finite non-empty set of the agent's *actions*. System joint action is a vector containing actions for all agents, and $ACS = ACA_{ag_1} \times \ldots \times ACA_{ag_n}$ is a set of all joint actions. We use $acs\,[ag] \in ACA_{ag}$ notation to identify the contribution of agent ag to a system joint action.
- I_{ag} is a finite non-empty set of *internal states*.
- $act_{ag} : I_{ag} \to ACA_{ag}$ is a *decision function*, which defines which action agent performs in which internal state.
- $see_{ag} : S \times ACS \to 2^S \times 2^{ACS}$ is a *perception function*, which is used to model incomplete information.[1]
- $refn_{ag} : I_{ag} \times 2^S \times 2^{ACS} \to I_{ag}$ is a *state refinement function*, which defines the agent's internal state transitions depending on the agent's *perception* of a new environment state and a system joint action $acs \in ACS$.

A group of agents, which cooperate to achieve common goals, is called *coalition* and defined as a subset of the set of all agents $A \subseteq AG$ (including empty set $\emptyset$ and the whole set AG). We define a set of *joint states* of a coalition $A = \{ag_{j_1}, \ldots, ag_{j_k}\} \subseteq AG$ as a set of state vectors containing states for all agents in coalition:

$$FS_A = I_{ag_{j_1}} \times \ldots \times I_{ag_{j_k}}. \tag{1}$$

[1] Each particular agent can have incomplete information about the current state and decisions, chosen by other agents. Thus, the agent can only suppose that the environment now in one of the possible states and other agents chose one of the possible actions.

We use $fs_A[ag_{j_l}] = (i_{ag_{j_1}}, \ldots, i_{ag_{j_k}})[ag_{j_l}] \triangleq i_{ag_{j_l}}$ notation to identify the contribution of agent ag_{j_l} to joint state fs_A.

Note that, if a coalition B is a strict subset of a coalition A ($B \subseteq A$), then it is possible to build a natural projection of a coalition's A joint state onto a coalition's B joint state by excluding states of agents not belonging to B. We use $fs_A|_B$ notation to refer to this projection.

Joint decision function $act_A : FS_A \to 2^{ACS}$ for a coalition A in a joint state $fs_A \in FS_A$ is defined as follows:

$$act_A(i_{ag_{j_1}}, \ldots, i_{ag_{j_k}}) \triangleq \{acs \in ACS \,|\forall ag \in A : acs[ag] = act_{ag}(i_{ag})\}. \quad (2)$$

Thus, decisions of agents belonging to the coalition A are defined by their decision functions, and decisions of other agents are not restricted.

Joint perception of the coalition $see_A : S \times ACS \to 2^S \times 2^{ACS}$ is defined as follows:

$$see_A(s, a) \triangleq \bigcap_{ag \in A} see_{ag}(s, a). \quad (3)$$

Thus, joint coalition perception is more precise than a perception of an individual agent and, therefore, a coalition can make more accurate decisions.

Joint state refinement function $refn_A : FS_A \times S \times ACS \to FS_A$ is defined as follows:

$$refn_A\left((i_{ag_{j_1}}, \ldots, i_{ag_{j_k}}), s, a\right) \triangleq \begin{pmatrix} refn_{ag_{j_1}}(i_{ag_{j_1}}, see_A(s, a)), \\ \ldots, \\ refn_{ag_{j_k}}(i_{ag_{j_k}}, see_A(s, a)) \end{pmatrix}. \quad (4)$$

$MASL$ is based on the classical propositional logic, whose formulas specify environment state properties, and it extends this logic in two ways: Firstly, to specify environment behavior, we introduce a restricted variant of PDL; secondly, to specify system properties, we introduce an extended variant of ATL which includes the environment specification as a part of the model for the interpretation of formulas.

Let $Prop$ be a finite alphabet of propositions. The syntax of classical propositional logic language L_{es} is defined by the following grammar:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi, \quad (5)$$

where $p \in Prop$.[2]

A model for propositional logic is a pair $M_{es} = \{S, \pi\}$, where S is a set of environment states, and $\pi : S \to 2^{Prop}$ is a valuation function, which says for each state $s \in S$ which propositions are true in s. The semantics of the language are given via the satisfaction relation, "$\models_{es}$", which holds between pairs of the form M_{es}, s, (where M_{es} is a model, and $s \in S$ is an environment state), and formulae of the language. The semantic rules defining this relation are as follows:

[2] Here and further we define syntax and semantics only for the $\neg$ and $\wedge$ logical connectives. The remaining connectives of propositional logic can be defined as abbreviations in the usual way.

1. $M_{es}, s \vDash_{es} p$ iff $p \in \pi(s)$.
2. $M_{es}, s \vDash_{es} \neg\phi$ iff $M_{es}, s \nvDash_{es} \phi$.
3. $M_{es}, s \vDash_{es} \phi \wedge \psi$ iff $M_{es}, s \vDash_{es} \phi$ and $M_{es}, s \vDash_{es} \psi$.

Since the rules for the interpretation of propositions and the logic connectives $\neg$ and $\wedge$ are standard, we omit them in further semantics definitions.

2 Environment Behavior Specification

In addition to a set of states S, the multiagent system environment is described by relation $env \subseteq S \times ACS \times S$, which determines possible environment state transitions depending on system joint actions. Relation env is *serial* if for all $s \in S$ and $acs \in ACS$ there exists $s' \in S$ such that $(s, acs, s') \in env$. A set of possible results of a joint action acs in a state s is denoted $env(s, acs) \triangleq \{s' \in S \mid (s, acs, s') \in env\}$.

Let *Actions* be a finite alphabet of action symbols. The syntax of joint action specification language L_{ac} is defined by the following grammar:

$$\alpha ::= aca \mid \alpha \wedge \alpha \mid \neg\alpha, \tag{6}$$

where $aca \in Actions$, and the syntax of environment behavior specification language L_{env} is as follows:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid [\alpha]\phi, \tag{7}$$

where $p \in Prop$ and $\alpha \in L_{ac}$.

Thus, we extend the classical propositional logic (5) with a new syntactical construct $[\alpha]\phi$, which is interpreted as: "If a system performs a joint action for which α is true, then ϕ will be true for the next environment state".

A model for L_{ac} is a pair $M_{ac} = \{ACS, \theta\}$, where $ACS = ACA_{ag_1} \cup \ldots \cup ACA_{ag_n}$ is a set all joint actions, and $\theta : Actions \to ACA_{ag_1} \cup \ldots \cup ACA_{ag_n}$ is a valuation function. The semantics of L_{ac} are presented via the satisfaction relation, "$\vDash_{ac}$", which holds between pairs of the form M_{ac}, acs, (where M_{ac} is a model, and $acs \in ACS$ is a joint action), and formulae of the language. The semantic rules defining this relation include the rules for logical connectives and the following rule:

– $\theta, acs \vDash_{ac} aca$ iff there exists $ag \in AG$ such that $acs\,[ag] = \theta(aca)$.

This rule describes the interpretation of a single action symbol, and its intuitive meaning is "aca is true for the system joint action acs if there exists agent ag whose contribution to the joint action is $\theta(aca)$".

A model for L_{env} is a tuple $M_{env} = \{S, \pi, ACS, \theta, env\}$, where S and π as in M_{es}, ACS and θ as in M_{ac}, and $env \subseteq S \times ACS \times S$ is a relation which describes environment behavior. The semantics of L_{env} are given via satisfaction relation, "$\vDash_{env}$", which holds between pairs of the form M_{env}, s, (where M_{env} is a model, and $s \in S$ is an environment state), and formulae of the language. The semantic rules defining this relation include the rules for propositions and logical connectives, and the following rule:

– $M_{env}, s \vDash_{env} [\alpha]\phi$ iff forall $acs \in ACS$, if $\theta, acs \vDash_{ac} \alpha$ then forall $s' \in env(s, acs)$, we have $M_{env}, s' \vDash_{env} \phi$.

For the specification of joint actions, the model-checking problem is formulated as follows: for a given model $M_{ac} = \{ACS, \theta\}$ and a formula $\alpha \in L_{ac}$ find the maximal subset of joint actions $[\alpha] \subseteq ACS$ such that for all $acs \in [\alpha]$, we have $acs \vDash_{ac} \alpha$. Since the joint action specification logic is indeed a classical propositional logic, the model-checking problem is decidable in time $O(|\alpha| \cdot |ACS|)$, where $|\alpha|$ is the size of formula α and $|ACS|$ is the size of the set of all joint actions ACS.

For the specification of environment behavior, the model-checking problem is formulated as follows: for a given model $M_{env} = \{S, \pi, ACS, \theta, env\}$, a subset of the environment states $T \subseteq S$ and a formula $\phi \in L_{env}$ find the maximal subset of states $[\phi]_T \subseteq T$ such that for all $s \in [\phi]_T$, we have $M_{env}, s \vDash_{env} \phi$. Since the environment behavior specification logic is a subset of propositional dynamic logic, the complexity of the model checking is bounded by $O(|\phi| \cdot |ACS| \cdot |S|)$.

3 Multiagent System Specification

In this section, we propose a logic, whose syntax is similar to ATL syntax, and which allows formal specification of properties of the multiagent system model, defined in Section 1. This model acts in the environment, whose behavior satisfies *environment behavior specification* $\widetilde{env} \in L_{env}$.

Let $Coal$ be a finite alphabet of *coalition symbols*. The syntax of multiagent system specification language L_{MAS} is defined as follows:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \langle\langle A\rangle\rangle \bigcirc \phi \mid \langle\langle A\rangle\rangle \phi \, \mathcal{U}_{\leq t} \phi \mid \langle\langle A\rangle\rangle \Box_{\leq t} \phi, \tag{8}$$

where $p \in Prop$, $A \in Coal$, and $t \in \mathbb{N} \cup \{\infty\}$. Thus, the language of multiagent system specification includes the $\langle\langle A\rangle\rangle$ quantifier, whose intuitive meaning is "In any environment which satisfies $\widetilde{env}$, behavior of the coalition A according to coalition agents' programs results in ...", and the three temporal operators:

– $\bigcirc\phi$, intuitively "in the next environment state ϕ will be true".
– $\phi \, \mathcal{U}_{\leq t} \psi$, intuitively "the environment will reach a state where ψ holds in not more than t steps (eventually if $t = \infty$) visiting only states where ϕ holds".
– $\Box_{\leq t}\phi$, intuitively "the environment will be in states where ϕ holds for not less than t steps (always if $t = \infty$) including current moment".

Using the base temporal operator $\langle\langle A\rangle\rangle \phi \, \mathcal{U}_{\leq t} \psi$, we introduce a derived operator $\langle\langle A\rangle\rangle \Diamond_{\leq t} \phi \triangleq \langle\langle A\rangle\rangle true \, \mathcal{U}_{\leq t} \phi$, whose intuitive meaning is "the environment will reach a state where ψ holds in not more than t steps (eventually if $t = \infty$)".

We define a set of all *possible outcomes* for a coalition A in an initial joint state $fs_A \in FS_A$ and an environment state $s \in S$ as a set of sequences of triples:

$$out_A(fs_A, s) = \{\lambda \in (FS_A \times S \times ACS \cup \{\emptyset\})^+\}, \tag{9}$$

where for each sequence $\lambda \in out_A(fs_A, s)$, we have $\lambda[0] = (fs_A, s, \emptyset)$ and for each $0 < i < |\lambda|$, we have $\lambda[i]|_{ACS} \in act_A(\lambda[i-1]|_{FS_A})$ and $\lambda[i]|_{FS_A} = refn_A(\lambda[i-1]|_{FS_A}, \lambda[i]|_S, \lambda[i]|_{ACS})$, and there exists $env_\lambda \subseteq S \times ACS \times S$ such that $\bigcup_{j=1}^{|\lambda|-1} \{(\lambda[j-1]|_S, \lambda[j]|_{ACS}, \lambda[j]|_S)\} \subseteq env_\lambda$ and for each $0 \leq k < |\lambda|$, we have $\{env_\lambda, \pi, \theta\}, \lambda[k] \vDash \widetilde{env}$.[3] Thus, each element of sequence λ consists of:

- A coalition A joint state $fs_A \in FS_A$.
- An environment state $s \in S$.
- A system joint action $acs \in ACS \cup \{\emptyset\}$ (or $\emptyset$ for the first element).

Furthermore, actions and internal state transitions of the coalition agents are determined by their programs (act_A and $refn_A$) and environment state transitions satisfy the environment specification $\widetilde{env}$.

A model for L_{MAS} is a tuple $M = \{AG, S, \pi, \theta, \widetilde{env}, \gamma\}$, where:

- $AG = \{ag_1, \ldots, ag_n\}$ is a finite non-empty set of agents.
- S is a finite non-empty set of environment states.
- $\pi : S \to 2^{Prop}$ is a valuation function, which says for each state $s \in S$ which propositions are true in s.
- $\theta : Actions \to ACA_{ag_1} \cup \ldots \cup ACA_{ag_n}$ is a valuation function for action symbols.
- $\widetilde{env} \in L_{env}$ is a specification of environment.
- $\gamma : Coal \to 2^{AG}$ is a valuation function for coalition symbols.

The semantics of L_{MAS} are given via satisfaction relation, "$\vDash$", which holds between triples of the form $M, A, (fs_A, s)$, (where M is a model, $A \subset A$ ia a coalition, and $(fs_A, s) \in FS_A \times S$ is a pair of states), and formulae of the language. Intuitively, behavior of the coalition A in the initial joint state fs_A and the environment state s satisfies ϕ. The semantic rules defining this relation include the rules for propositions and logical connectives, and the following rules:

1. $M, A, (fs_A, s) \vDash \langle\langle B\rangle\rangle \bigcirc \phi$ iff $\gamma(B) \subseteq A$ and forall $\lambda \in out_{\gamma(B)}(fs_A|_{\gamma(B)}, s)$, we have $|\lambda| > 1$ implies $M, \gamma(B), \lambda[1]|_{FS_A \times S} \vDash \phi$.
2. $M, A, (fs_A, s) \vDash \langle\langle B\rangle\rangle \phi \, \mathcal{U}_{\leq t} \psi$ iff $\gamma(B) \subseteq A$ and there exists finite $0 \leq i \leq t$ such that forall $\lambda \in out_{\gamma(B)}(fs_A|_{\gamma(B)}, s)$, we have $|\lambda| > i$ implies that there exists $0 \leq k \leq i$ such that $M, \gamma(B), \lambda[k]|_{FS_A \times S} \vDash \psi$ and forall $0 \leq j < k$, we have $M, \gamma(B), \lambda[j]|_{FS_A \times S} \vDash \phi$.
3. $M, A, (fs_A, s) \vDash \langle\langle B\rangle\rangle \Box_{\leq t} \phi$ iff $\gamma(B) \subseteq A$ and forall $\lambda \in out_{\gamma(B)}(fs_A|_{\gamma(B)}, s)$ and forall $0 \leq i \leq \min(t, |\lambda| - 1)$, we have $M, \gamma(B), \lambda[i]|_{FS_A \times S} \vDash \phi$.

Note that, in the semantics definition above, there is no such ATL notion as *strategy*. Thus, although $MASL$ syntax is similar to ATL syntax, $MASL$ semantics is closer to $TCTL_S$. What sets the $MASL$ semantics apart from $TCTL_S$ and

[3] We use $\lambda[i]$ notation to refer i-th element of the sequence λ, $\lambda[i]|_S$, $\lambda[i]|_{FS_A}$ and $\lambda[i]|_{ACS}$ notation to refer particular part of the $\lambda[i]$ and $|\lambda|$ notation to refer the length of the sequence λ.

ATL is that, during the interpretation of a sub-formula $\langle\langle A\rangle\rangle \ldots$, the model and current state are constricted with respect to coalition $\gamma(A)$. Also, note that it is possible to constrict the model and state, but it is not possible to extend them.

For $MASL$ specifications,the model-checking problem is formulated as follows: for a given model $M = \{AG, S, \pi, \theta, \widetilde{env}, \gamma\}$, a coalition A, a set of pairs $T = \{(fs_A, s) \in FS_A \times S\}$ and a formula ϕ find the maximal subset $[\phi]_T \subseteq T$ such that for all $(fs_A, s) \in [\phi]_T$, we have $M, A, (fs_A, s) \vDash \phi$. The complexity of the task is bounded by a polynomial in:

- The length of formula $|\phi|$.
- The length of environment specification $|\widetilde{env}|$.
- The size of the set of all joint states for the coalition of all agents $|FS_{AG}|$
- The size of the set $|S \times ACS \times S|$ in a power n, where n is the maximal depth of nesting of the [ ] operator in the environment specification $\widetilde{env}$.

If $n = 1$, the model-checking complexity of $MASL$ is polynomial in the length of formula $|\phi|$, size of the set of joint sates for the coalition of all agents $|FS_{AG}|$, size of the set of environment states $|S|$ and size of the set of joint actions $|ACS|$.

4 Conclusions

The problem of building a formal method for specifying multiagent systems receives increasing attention of the science community. Two recent works on Coalition Logic [10] and Alternating-time Temporal Logic [11] made a significant breakthrough in this area by introducing explicit approaches to reasoning about the cooperation and competition of agents. In [17] $PDL+CL$ logic is introduced to reason about interaction between agents and nondeterministic environment, as well as to describing coalition ability to achieve a certain state of affairs.

In this paper we have introduced a new logic, $MASL$, which is based on the principles which are similar to those of $PDL+CL$. $MASL$ allows explicit specification of the complex temporal properties, including reaction time constraints, and automated system verification via model-checking.

To reason about complex temporal properties, we have utilized syntax, which is similar to ATL syntax, and proper semantics. To enable automated verification we have extended the intelligent agent model from [19] to deal with the case of multiple agents and proposed the logic for the specifying properties of the model. To specify the reaction time constraints, we have used subscript index construct by analogy with [20].

We have made estimations on the model-checking complexity and identified the case when the polynomial model-checking is possible. As far as we concern, the restricted variant of the logic which allows polynomial model checking can be used as a base for a formal method of specification and verification of multiagent real-time systems. It also can used as a goal definition language for a multiagent planning algorithm with temporally extended goals.

Our future work will focus on implementing the $MASL$ symbolic model-checking by using available decision diagrams package, for example, $CUDD$ [21],

on developing the logic by adding means of reasoning about agents' beliefs, desires and intensions, and on extending the application domain of the logic by introducing planning and system synthesis algorithms.

References

1. Jenings, N., Wooldridge, M.: Intelligent Agents. In: Weiss, G. (ed.) Multiagent Systems, pp. 377–421. MIT Press, Cambridge (2001)
2. Jenings, N., Wooldridge, M.: Applications of intelligent agents (2000)
3. Parunak, H.D.: Industrial and Practical Application of DAI. In: Weiss, G. (ed.) Multiagent Systems, pp. 377–421. MIT Press, Cambridge (2001)
4. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science. Formal Models and Sematics (B), vol. B, pp. 995–1072. North-Holland Pub. Co., Amsterdam (1990)
5. van Langevelde, I., Philipsen, A., Treur, J.: Formal specification of compositional architectures. In: ECAI '92. Proceedings of the 10th European conference on Artificial intelligence, New York, NY, USA, pp. 272–276. John Wiley & Sons, Inc., Chichester (1992)
6. Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J.: DESIRE: Modelling multi-agent systems in a compositional formal framework. Int. Journal of Cooperative Information Systems 6(1), 67–94 (1997)
7. Fisher, M.: A survey of Concurrent MetateM - the language and its applications. In: Gabbay, D.M., Ohlbach, H.J. (eds.) ICTL 1994. LNCS, vol. 827, pp. 480–505. Springer, Heidelberg (1994)
8. Spivey, J.M.: The Z notation: a reference manual. Prentice Hall International (UK) Ltd., Hertfordshire, UK (1992)
9. d'Inverno, M., Luck, M.: Understanding autonomous interaction. In: ECAI '96: European Conference on Artificial Intelligence, pp. 529–533 (1996)
10. Pauly, M.: Logical for Social Software. PhD thesis, University of Amsterdam (2001)
11. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(1), 672–713 (2002)
12. van der Hoek, W., Wooldridge, M.: Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. Studia Logica 75(1), 125–157 (2003)
13. Pauly, M., Wooldridge, M.: Logic for mechanism design — a manifesto. In: GTDT-2003: Proceedings of the 2003 Workshop on Game Theory and Decision Theory in Agent Systems (2003)
14. Ryan, M., Schobbens, P.-Y.: Agents and roles: refinement in alternating-time temporal logic. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 100–114. Springer, Heidelberg (2002)
15. Agotnes, T., van der Hoek, W., Wooldridge, M.: On the logic of coalitional games. In: AAMAS '06. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 153–160. ACM Press, New York (2006)
16. Agotnes, T., van der Hoek, W., Wooldridge, M.: Temporal qualitative coalitional games. In: AAMAS '06. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 160–167. ACM Press, New York (2006)

17. Sauro, L., Gerbrandy, J., van der Hoek, W., Wooldridge, M.: Reasoning about action and cooperation. In: AAMAS '06. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 185–192. ACM Press, New York (2006)
18. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. SIGACT News 32(1), 66–69 (2001)
19. Bugaychenko, D.Y., Soloviev, I.P.: Abstract architecture of an inteligent agent. Software Enginering 1, 36–67 (2005)
20. Laroussinie, F., Schnoebelen, P., Turuani, M.: On the expressivity and complexity of quantitative branching-time temporal logics. Theor. Comput. Sci. 297(1-3), 297–315 (2003)
21. Somenzi, F.: CUDD: Colorado University Decision Diagram package, `http://vlsi.colorado.edu/fabio/CUDD/`

Modeling of Agents in Organizational Context

Alexei Sharpanskykh

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands
sharp@few.vu.nl

Abstract. At present the agent paradigm is often used for computational modeling of human behavior in an organizational setting. However, not many of the existing computational approaches make use of a rich theoretical basis developed in social science. Therefore, often mathematically sound models are invalid in practice. This paper proposes a formal approach for modeling of characteristics and behavior of agents in organizations, diverse aspects of which are represented using an expressive formal framework. The approach is based on the theoretical findings from social science and enables analysis of how different organizational and environmental factors influence the behavior and performance of agents. The approach is illustrated by a simulation case study.

1 Introduction

The agent paradigm has been extensively used for modeling and analysis of both human and artificial organizations. In particular, in the area of Multi-Agent Systems (MAS) artificial agent organizations have been investigated [4, 9, 15]. Representation of MAS as an organization consisting of roles and groups facilitates handling high complexity and poor predictability of dynamics in a system [9]. Some of the MAS frameworks are also aimed at modeling human organizations [6, 8, 16]. Although such models can be computationally effective, most of them lack the expressivity for conceptualizing a wide range of concepts and relations of human organizations. Also, such frameworks only rarely use extensive theoretical findings from social science.

Modeling of individuals in a social setting using the agent paradigm has gained popularity in the area of computational social science [2]. In this area many approaches have been developed for analyzing, predicting and improving the efficiency of task allocation to and execution by agents. In particular, the frameworks TAEMS [5] and VDT [12] provide the elaborated models for task environments and the computational means to analyze the performance of agents and of a whole MAS with respect to the task execution. In [21] it is demonstrated how agents can be allocated to tasks based on the match between the task descriptions and the agent capabilities. The agents in these and other similar frameworks are represented as autonomous entities characterized by skills, competences, experience, and, sometimes, goals. In task-oriented agent-based modeling it is often assumed that agents comply with organizational goals and will perform tasks in such a way that a high level of organizational performance is ensured. However, in some cases such an assumption may not be valid. In particular, for feasible modeling of human organizations various (sometimes conflicting) interests of different organizational

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 193 – 203, 2007.

actors should be explicitly considered, as they often (significantly) influence the organizational performance. In general, to be productive an organization should arrange work and provide incentives to its employees in such a way that they are constantly motivated to adopt the behavior that ensures the satisfaction of the essential organizational goals. The topic of work motivation received much attention in Organization Theory [14,17]. Also, in the area of MAS different motivation models and the mechanisms for manipulating them have been proposed [3]. However, only a little research has been done on the computational modeling of motivation of agents situated in organizational context. Organizational factors that influence motivation of agents are diverse and often interrelated: e.g., norms and regulations related to the task execution, to communication, a power system, a reward/punishment system etc.

In this paper, a formal agent-based approach for modeling of characteristics and behavior of individuals in organizational context is proposed. The approach makes use of a rich theoretical basis developed in Organization Theory. In particular, the motivation modeling of agents is based on the expectancy theory (the version of Vroom) [23] that has received good empirical support. The formal motivation modeling has an advantage that automated tools can be developed using which managers can make estimations of how different organizational factors influence the motivation and performance of different types of employees (agents). Agents are situated in a formal organization modeled using the general organization modeling and analysis framework proposed in [10]. This framework comprises several interrelated views: the performance-oriented view [19] describes organizational and individual goal and performance indicators structures; the process-oriented view [18] describes task and resource structures and dynamic flows of control; within the organization-oriented view [10] organizational roles, their power and communication relations are defined. Concepts and relations within every view, as well as models of agents are formally described using dedicated languages based on an order sorted predicate logic [13]. Temporal aspects of agent-based organizational models are formalized using the Temporal Trace Language (TTL) [22], which is an extension of a sorted predicate logic that allows reasoning about dynamic properties of systems.

The paper is organized as follows. Section 2 introduces the proposed modeling approach. The application of the approach is illustrated by a simulation case study in Section 3. Section 4 concludes the paper.

2 An Agent-Based Modeling Approach

In a formal organizational model goals are structured into a hierarchy using the refinement relations. Goals are satisfied by execution of tasks. Different sets of tasks are associated with roles. Interaction and authority structures are defined on roles with respect to tasks. To enable effective and efficient execution of tasks, agents with appropriate characteristics should be allocated to roles. In this Section, a description of different types of agent characteristics is provided (Section 2.1), followed by the introduction of a motivation model of an agent (Section 2.2).

2.1 Characteristics of Agents and Allocation to Roles

For each role a set of requirements on agent *capabilities* (i.e., knowledge and skills) and *personal traits* is defined. Requirements on knowledge define facts and

procedures, confident understanding of which is required from an agent. Skills describe developed abilities of agents to use effectively and readily their knowledge for tasks performance. In literature [7] four types of skills relevant in organizational context are distinguished: technical (related to the task's specific content), interpersonal (e.g., communication, cooperation), problem-solving/decision-making and managerial skills. More specific requirements may be defined on skills reflecting their level of development, experience, the context in which these skills were attained. To enable testing (or estimation) of skills and knowledge, every particular skill and knowledge is associated with a performance indicator(s) (PI) (e.g., the skill 'typing' is associated with the PI "the number of characters per minute"). Moreover, a skill may be associated with a compound PI built as a weighed expression on simple PIs.

In psychology [11] personal traits that may also influence the execution of tasks are divided into five broad categories: openness to experience, conscientiousness, extroversion, agreeableness, and neuroticism. Some agent's traits may mediate the attainment of agent's skills. For example, extroversion and agreeableness play an important role in building interpersonal skills.

Agent capabilities and traits can have different levels of importance. Whereas the required for a role capabilities and traits are compulsory for taking the role, desired capabilities and traits considered as an advantage.

In modern social science behavior of individuals is considered as goal-driven. A goal is as an objective to be satisfied describing a desired state or development of the individual. It is recognized that high level goals of individuals are dependant on their needs. Currently the following division of needs is identified in social science: (1) *extrinsic needs* associated with biological comfort and material rewards; (2) *social interaction* needs that refer to the desire for social approval, affiliation and companionship; (3) *intrinsic needs* that concern the desires for self-development, self-actualization, and challenge. The discussed characteristics of an agent can be formalized using the sorted first-order predicate logic as it will be shown in Section 3.

In general, the efficiency of allocation of an agent to a role is dependant on how well the agent's characteristics fit with the role requirements. However, modern organizations implement very diverse allocation principles (e.g., based on equality, seniority or stimulation of novices) [17]. Such principles can be formalized as allocation policies comprising executable (temporal) rules (see Section 3).

In modern organizations when an individual is allocated to a role, the identification of his/her specific lower level goals is performed in cooperation with a managerial representative of the organization. During this process, the high level goals, based on the agent's needs are refined into more specific goals aligned with organizational goals using AND- and OR- relations as it is shown in [19]. Many authors argue that the lower level goals should as detailed and specific as possible [7, 17]. Often two types of such goals are distinguished: development (or learning) and performance goals. Development goals reflect wishes of agents to gain certain knowledge or some skills that are also useful for the organization. Individuals vary in the abilities and desires to learn; therefore, this type of goals is particularly dependent on the individuals' traits and goals. Performance goals usually concern the effectiveness and efficiency of the execution of the tasks already allocated to the agent. Both development and performance goals may change over time.

Within the performance-oriented view of the modeling framework [19] the formal specification of a goal is based on a mathematical expression over a PI(s). The

characteristics of a goal include, among others: *priority*; *horizon* – for which time point/interval should the goal be satisfied; *hardness* – hard (satisfaction can be established) or soft (satisfaction cannot be clearly established, instead degrees of *satisficing* are defined); *negotiability*. For example, the hard performance goal "it is required to maintain the time for the generation of a plan < 24 hours" is based on the PI "the time for the generation of a plan". Another example is the development goal "it is desired to achieve the state in which the framework JADE is mastered". In the latter example the goal is desirable, which points at its low priority.

The satisfaction of goals in the organizational context is associated directly or indirectly with the performance of tasks. In particular, goals associated with intrinsic needs are often satisfied by intrinsic rewards that are a natural consequence of the agent behavior related to the execution of a task. Some agents receive intrinsic rewards irrespectively of the execution results. While intrinsic rewards for other agents are contingent upon the execution outcomes. In the latter case if the actual task result equates to or exceeds the agent's expectation, then the agent receives an intrinsic reward. Furthermore, as follows from [17] the amount of intrinsic reward is dependent on the task complexity. Externally provided rewards (e.g., salary, bonuses, group acceptance) serve to the satisfaction of goals related to extrinsic and social interaction needs. At any time point the (level of) satisfaction of a lower level goal may be established by the evaluation of the PI expression, on which the goal is based. Further, using the rules defined in [19] information about the satisfaction of lower-level goals is propagated to determine the satisfaction of high-level goals.

Many organizations have reward/sanction systems contingent on the satisfaction of goals. Furthermore, besides general organizational policies also particular individual policies (e.g., concerning promotions, bonuses etc.) can be defined. Many studies showed that making explicit rules based on which rewards and sanctions are provided increases the motivation of an agent to perform certain actions (tasks) [7]. The motivation of agents to perform certain tasks is important to ensure the satisfaction of both individual and organizational goals related (directly or indirectly) to these tasks. Therefore, the motivational aspect of the agent behavior should be explicitly represented in the models of organizational agents.

2.2 Modeling the Motivation of an Agent

The topic of motivation in work organizations has received much attention in social science [7, 14, 17, 23]. In [23] the motivation is defined as a *process governing choice made by persons among alternative forms of voluntary activity*. We adopt the Vroom's version of the expectancy theory [23] that received good empirical support.

According to this theory, when an individual evaluates alternative possibilities to act, s/he explicitly or implicitly makes estimations for the following factors: *expectancy*, *instrumentality*, and *valence* (see Fig.1).

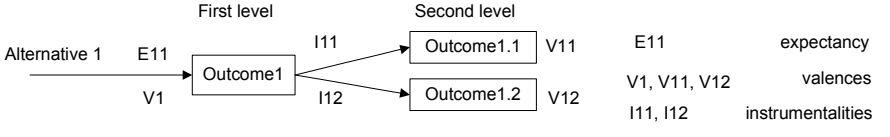


Fig. 1. An example of the motivation model by Vroom [23]

Expectancy refers to the individual's belief about the likelihood that a particular act will be followed by a particular outcome (called a first-level outcome). In organizational context expectancy of an agent related to successful task execution is determined by the characteristics of the task and the agent, and by the organizational and environmental conditions. Tasks can be characterized along multiple dimensions: (a) complexity and predictability; (b) specialization; (c) formalization. Usually agents that possess knowledge and the skills required for some task have a high level of expectancy of the successful task execution. Furthermore, agents with highly developed skills tend to assign a high value to expectancy associated with complex and not completely predictable tasks. On the opposite, inexperienced agents decrease their expectancy when dealing with complex tasks. For such agents the formalization of a task (e.g., by detailed procedure descriptions) will increase their expectancy level. If a task requires from an agent a contribution from or collaboration with other agents, then the agent's belief about reliability of these agents will play an important role in his/her expectancy estimation. Many modern organizations actively interact with the environment, which is often highly dynamic and unpredictable. The less certainty and knowledge about the environment an agent has, the less his/her expectancy level. As expectancy is defined as a subjective perception (or a belief) of an agent, the agent's personal traits also have influence on his/her expectancy.

Instrumentality is a belief concerning the likelihood of a first level outcome resulting into a particular second level outcome; its value varies between -1 and +1. A second level outcome represents a desired (or avoided) by an agent state of affairs that is reflected in an agent's goal(s) (e.g., bonus receipt, imposition of a sanction). Although the notion of instrumentality can be perceived as probability, in contrast to the latter instrumentality may take negative values, in case a second-level outcome does not follow a particular first-level outcome. Note that the agent's experience gained by the execution of tasks influences the values of expectancies and instrumentalities associated with these tasks. For example, if despite high performance the agent did not get the promised/expected (amount of) rewards, then his/her instrumentality between the level of efforts and the previously identified reward will decrease. Similarly, the agent adjusts the expectancy value associated with a task based on his/her actual amount of efforts put into the task execution.

Valence refers to the strength of the individual's desire for an outcome or state of affairs. While second level outcomes are directly related to the agent's goals, the valence values associated with these outcomes refer to priorities of these goals. Thus, similarly to goal priorities, the values of valence change over time (e.g., depending on the satisfaction of goals).

In the Vroom model *the force on an individual to perform an act is a monotonically increasing function of the algebraic sum of the products of the valences of all outcomes and the strength of his expectancies that the act will be followed by the attainment of these outcomes* [23]: $F_i = f\left(\sum_{j=1}^{n} E_{ij} \times V_j\right), \quad V_j = \sum_{k=1}^{m} V_{jk} \times I_{jk}$ here E_{ij} is the strength of the expectancy that act i will be followed by outcome j; V_j is valence of first-level outcome j; V_{jk} is valence of second-level outcome k that follows first-level outcome j; I_{jk} is perceived instrumentality of outcome j for the attainment of outcome k.

3 A Simulation Case Study

In this Section we shall investigate the behavior of the employees of a small firm that develops web graphics by request from external clients. Such an organization manages all its activities using a cohesive team structure. Teams have a flat power structure. Although the role of a leader (or manager) is identified, all important decisions are made with the assistance of all team members. The manager is responsible mostly for organizing tasks: e.g., searching for clients, distribution of orders, monitoring of the order execution. The firm consists of highly motivated members and has a very informal and open working style. The risky, environment-dependant nature of the firms of such type may cause financial insecurity and deficiency for their members. In the following the model used for the simulation is introduced. Due to the space limitation the model introduction will be mostly informal, providing the formalization only for the most essential parts. Subsequently, the simulation results are presented.

Modeling tasks and the environment

Tasks received by the firm are characterized by: (1) *name*; (2) *type*; (3) *required level(s) of development of skill(s)*; (4) *average / maximum duration*; (5) *extra time delay per unit of each skill development*; (6) *material reward*; (7) *intrinsic reward*; (8) *development level increment per skill*. For this case study the generalized PI "the development level" for each skill is used, which is an aggregated quantity (a real number in the range 0-5) reflecting the skill-related knowledge, experience, task execution context etc. The task average duration is the time that an agent that possesses the skills satisfying the task requirements will spend on the task execution. Agents with insufficient development levels of skills will require additional time for the execution. This is specified by the extra time delay characteristic per deficient unit of each required skill. The maximum task duration specifies the maximal time allowed for the task execution. For the successful performance of tasks agents are granted with material rewards; also the development level(s) of their skill(s) is (are) increased by the experience increment amount(s). Note that for simplicity the intrinsic rewards associated with the tasks in this case study are made independent of the specific characteristics of the agents who execute these tasks. The task types used in the simulation are specified in Table 1.

In the simulation we suppose that tasks arrive in accordance with a non homogeneous Poisson process $\{N(t), t \geq 0\}$ with a bounded intensity function $\lambda(t)$. Here $N(t)$ denotes the number of events that occur by time t and the quantity $\lambda(t)$ indicates how likely it is that an event will occur around the time t. We use the thining or random sampling approach [20], which assumes that $\lambda(t) \leq \lambda$ for all $t \leq T$, where T is the simulation time (2000 working hours (1 year) for this case study). Furthermore, for $T \leq 1000$: $\lambda_{A1}=\lambda_{A2}=\lambda_{B1}=\lambda_{B2}=0.05$ and for $T > 1000$: $\lambda_{A1}=\lambda_{A2}=2 * 10^{-5}$; $\lambda_{B1}=\lambda_{B2}=0.05$.

Organization modeling

The firm has two high level long-term goals with the same priority: "it is required to maintain a high profit level" and "it is required to maintain a high level of satisfaction of the employees". These goals are imposed on the organizational structure that comprises the role Manager and the generalized role Task Performer. The latter is instantiated into

Table 1. The characteristics of the task types A1/A2 (create a simple/complex web illustration) and B1/B2 (create a simple/complex Flash animation) used in the simulation

Type	A1	A2	B1	B2
Required skill(s)	S1: 2	S1: 4	S2: 1	S2: 4
Average (max) duration (hours)	14 (18)	30 (38)	12 (15)	50 (60)
Extra time delay per skill (hours)	S1: 2	S1:4	S2: 3	S2: 8
Material reward	10	20	7	25
Intrinsic reward	1	3	1	4
Development increment	S1:0.1	S1:0.2	S2: 0.08	S2: 0.2

specific roles-instances associated with the tasks received by the firm. An instantiated role is assigned to one of the agents representing the employees using the following policy: Agents that can be potentially allocated to a role should be *qualified* for this role. An agent is qualified for a role under two conditions: (1) the agent is not involved into the execution of any other tasks; (2) agent possesses the skills required for the task associated with the role; and the level of development of these skills will allow to the agent to finish the task before the task deadline (i.e., maximum duration). To formalize these conditions, for each task and agent characteristic a predicate is introduced. Some of these predicates are given in Table 2. To express the temporal aspects of the agent qualification rule the language TTL is used [22]. TTL specifies the dynamics of a system by a trace, i.e. a temporally ordered sequence of states. Each state corresponds to a particular time point and is characterized by a set of state properties that hold in this state. State properties are defined as formulae in a sorted predicate logic using state ontologies. A state ontology defines a set of sorts or types (e.g., TASK, AGENT), sorted constants, functions and predicates (see Table 2). States are related to state properties via the satisfaction relation |=: state(γ, t) |= p, which denotes that state property p holds in trace γ at time t. Dynamic properties are specified in TTL by relations between state properties.

Table 2. Predicates for the formalization of agent-based models

Predicate	Description
task_arrived, task_started, task_finished: TASK	Specifies the arrival, start and finish of a task
role_for_task: ROLE x TASK	Identifies a role for a task
agent_allocated: AGENT x ROLE	Specifies an agent allocated to a role
agent_qualified_for: AGENT x ROLE	Specifies an agent qualified for a role

The agent qualification rule is formally expressed in TTL as follows:

$\forall\gamma\ \forall$t:TIME $\forall$a1:TASK $\forall$ag:AGENT $\forall$r1:ROLE $\forall$tp1: TASK_TYPE
state(γ, t)|= [task_arrived(a1) $\wedge$ role_for_task(r1, a1) $\wedge$ task_type(a1, tp1) $\wedge$
$\neg\exists$r2:ROLE r2$\neq$r1 $\wedge$ agent_allocated(ag, r2) $\wedge$ sum([sk:SKILL], $\exists$VALUE:n, m, k case(state(γ, t)|= task_requires_skill(a1, sk, n) $\wedge$ agent_possesses_skill(ag, sk, m) $\wedge$ m$\geq$0.5$\wedge$ task_extra_delay(tp1, sk, k), k $\bullet$ (n-m), 0)) < (task_max_duration(tp1) - task_average_duration(tp1))
$\Rightarrow\ \forall$t1:TIME t1>t state(γ, t1)|= agent_qualified_for(ag, r1)

Here in sum([summation_variables], case(logical_formula, value1, 0)) logical_formula is evaluated for every combination of values from the domains of each from the summation_variables; and for every evaluation when logical_formula is true, value1 is added to the resulting value of the sum.

Further, since the firm recognizes the importance of wishes of its employees, a role can be only allocated, when a qualified agent has voluntarily requested the role. Furthermore, the firm established the rule that in case several qualified agents requested a role, then the agent with the earliest previous allocation time among these agents will be allocated to the role. These rules are also formalized using TTL.

For the successful execution of tasks the agents are provided with material rewards on the following basis: 50% of the reward is given to the agent who performed the task and the rest is divided equally among all other employees.

Modeling agents

The firm consists of three members and the manager modeled as agents. As in the most firms of such type, the employees are intrinsically motivated by their work, and pursuit high performance standards. For each agent two high level long-term hard goals are defined that also comply with the organizational goals: g1: it is required to maintain the level of income not less than 50; g2: it is required to maintain the level of intrinsic satisfaction not less than 5.

Two agents ag1 and ag2 possess the skill S1 to perform purely graphical work: agent_possesses_skill(ag1, S1, 4) and agent_possesses_skill(ag2, S1, 3). Here the third argument denotes the level of the skill development. The agent ag3 has the skill S2 to make Flash animations: agent_possesses_skill(ag3, S2, 4). Furthermore, ag1 has the general knowledge related to S2 (agent_possesses_skill(ag1, S2, 0.1)), which however is insufficient for the performance of tasks that require S2. By mutual consent of the firm and ag1 the development goal for ag1 without a strict deadline has been set: it is desired to achieve the level of development of S2 ≥ 0.5. When ag1 decides to gain the minimum level of the skill S2 development that is necessary for the task execution (0.5), s/he will be given one week for the training, during which no other tasks will be assigned to him/her. The motivation of the agents to attain their goals is represented by the motivation models, two examples of which for ag1 are given in Fig. 2.

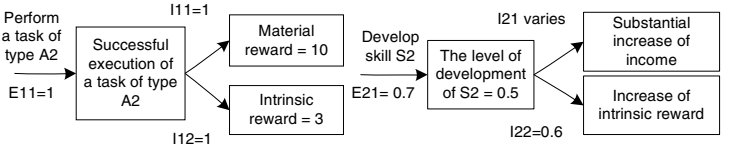


Fig. 2. The examples of two motivation models for the agent ag1 used in the case study

The parameters of the motivation models are defined as follows: Expectancy of an agent ag for the successful execution of a task tk is defined as a weighed average of the quotients $pos(sk_i)/req(sk_i)$ for each skill sk_i required for tk; here $pos(sk_i)$ is the development level of the skill sk_i possessed by ag and $req(sk_i)$ is the level required by tk. Instrumentality for each second level outcome associated with the successful execution of a task is equal to 1 for every agent qualified for this task. This is because the reward system is defined explicitly and the qualified agents have a clear estimation of the intrinsic reward associated with the task. The instrumentality value of ag1 for the skill S2 development is reevaluated in the end of each month and is equal to 1, when n/m > 50, and is equal to n/(m*50) otherwise; here n is the amount of the material rewards provided by the tasks of types B1 and B2 received by the firm up

to the current time point, and m is the amount of months of the simulation time (the initial instrumentality value is 0.35). The valence values of second level outcomes change over time. In particular, when the goal g1 of an agent ag is not satisfied, then the valence values of ag for all outcomes related to material rewards will become 1, and the valence values of outcomes related to intrinsic rewards will become 0.5. When g1 is satisfied, then the valence values for material outcomes will decrease to 0.5, and for intrinsic outcomes will increase to 1. An agent generates a request to perform an action specified in a motivation model (e.g., request for a role), when the motivational force associated with this action is greater than 0.5.

The initial income value is 20 for all agents, and the initial intrinsic satisfaction level is 3. Each agent consumes 0.05 units of the received material rewards per day and the amount of the received intrinsic rewards decreases by 0.03 each day.

The simulation is performed using the dedicated tool [1]. Fig. 3a shows how the motivational force of ag1 to attain the skill S2 changes over time. After the time point 1000, when the amount of tasks of type A diminishes significantly, the force transgresses the threshold 0.5, and ag1 begins the attainment of S2. After some time ag1 possesses the skills required to perform the tasks of both types A and B and both his/her goals g1 and g2 become satisfied (see Fig. 3b).

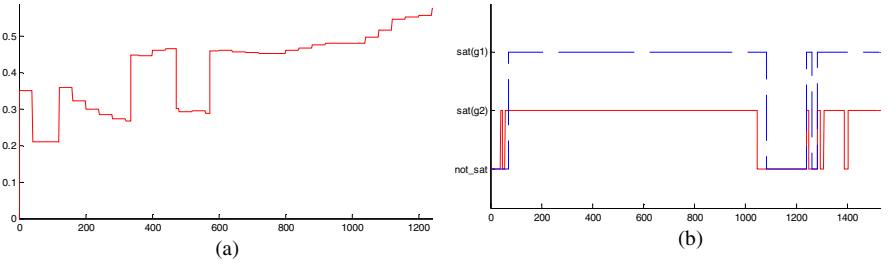


Fig. 3. (a) The change of the motivation force (the vertical axis) of agent ag1 for the attainment of skill S2 over time. (b) The change of the satisfaction of the goals of agent ag1 over time.

4 Conclusion

The paper proposes a formal approach for modeling of agents situated in an (formal) organization that accentuates the intentional and motivational aspects of agent behavior. The proposed quantitative motivation model of an agent based on the expectancy theory allows estimating the agent's motivational force to attain certain (organizational or individual) goals. Since the goal expressions are based on performance measurements, using the proposed approach it is possible to analyze how different organizational factors that affect the parameters of the motivation model influence the organizational or agent performance. An example of such analysis is demonstrated by a simulation case study in this paper.

Based on a large corpus of empirical social studies a great number of dependencies between organizational and environmental factors and the agent's motivation have been identified. These factors and dependencies stem from the different organizational views introduced above and will be further used for modeling and analysis of the behavior of agents situated in different types of organizations.

References

1. Bosse, T., Jonker, C.M., Meij, L., van der Treur, J.: A Language and Environment for Analysis of Dynamics by SimulaTiOn. Int. J. of Art. Intelligence Tools (to appear, 2007)
2. Carley, K.M.: A comparison of artificial and human organizations. Journal of Economic Behavior & Organization 31(2), 175–191 (1996)
3. Coddington, M., Luck, M.: A Motivation Based Planning and Execution Framework. International Journal on Artificial Intelligence Tools 13(1), 5–25 (2004)
4. Dastani, M., Hulstijn, J., Dignum, F., Meyer, J.-J.: Issues in Multiagent System Development. In: AAMAS'04. Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 922–929. ACM Press, New York (2004)
5. Decker, K.: TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms. In: O'Hare, G., Jennings, N. (eds.) Foundations of Distributed Artificial Intelligence, ch. 16, pp. 429–448. Wiley Inter-Science, Chichester (1996)
6. Erasmo, L., Montealegre Vazquez, M., Lopez y Lopez, F.: An Agent-Based Model for Hierarchical Organizations. In: Proceedings of COIN@AAMAS'06 workshop, pp. 32–47 (2006)
7. Galbraith, J.R.: Designing organizations. Jossey-Bass, San Francisco, California (1995)
8. Grossi, D., Dignum, F., Royakkers, L., Dastani, M.: Foundations of Organizational Structures in Multi-Agent Systems. In: AAMAS'05. Proceedings of Fourth International Conference on Autonomous Agents and Multiagent Systems, pp. 690–697. ACM Press, New York (2005)
9. Horling, B., Lesser, V.: A Survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(4), 281–316 (2005)
10. Jonker, C.M., Sharpanskykh, A., Treur, J., Yolum, P.: A Framework for Formal Modeling and Analysis of Organizations. Applied Intelligence 27(1), 49–66 (2007)
11. Katz, D., Kahn, R.: The social psychology of organizations. Wiley, New York (1966)
12. Kunz, J.C., Levitt, R.E., Jin, Y.: The Virtual Design Team: A computational simulation model of project organizations. Communications of the Association for Computing Machinery 41(11), 84–92 (1999)
13. Manzano, M.: Extensions of First Order Logic. Cambridge University Press, Cambridge (1996)
14. March, J.G., Simon, H.A.: Organizations. Wiley, New York (1958)
15. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: Proceedings of Agent-Oriented Software Engineering Workshop, pp. 185–193 (2000)
16. Partsakoulakis, I., Vouros, G.A.: Agent-enhanced Collaborative Activity in Organized Settings. International Journal of Cooperative Information Systems 15(1), 119–154 (2006)
17. Pinder, C.C.: Work motivation in organizational behavior. Prentice-Hall, Upper Saddle River, NJ (1998)
18. Popova, V., Sharpanskykh, A.: Formal Analysis of Executions of Organizational Scenarios based on Process-Oriented Models. In: Proceedings of the ECMS'07, pp. 36–44 (2007)
19. Popova, V., Sharpanskykh, A.: Formal Modeling of Goals in Agent Organizations. In: Proceedings of the Agent Organizations: Models and Simulations Workshop, pp. 74–86 (2007)
20. Ross, S.: Simulation, 2nd edn. Harcourt Academic Press, London, Boston, New York (1998)

21. Scholz, T., Timm, I.J., Spittel, R.: An Agent Architecture for Ensuring Quality of Service by Dynamic Capability Certification. In: Eymann, T., Klügl, F., Lamersdorf, W., Klusch, M., Huhns, M.N. (eds.) MATES 2005. LNCS (LNAI), vol. 3550, pp. 130–140. Springer, Heidelberg (2005)
22. Sharpanskykh, A., Treur, J.: Verifying Interlevel Relations within Multi-Agent Systems. In: Proceedings of European Conference on Artificial Intelligence, pp. 247–254. IOS Press, Amsterdam (2006)
23. Vroom, V.H.: Work and motivation. Wiley, New York (1964)

Motivations as an Abstraction of Meta-level Reasoning

Felipe Meneguzzi and Michael Luck

Department of Computer Science, King's College London, London WC2R 2LS, UK
{felipe.meneguzzi,michael.luck}@kcl.ac.uk

Abstract. In agent systems, meta-level reasoning is commonly used in enforcing rationality in the choice of goals and actions performed by an agent, ensuring that an agent behaves as effectively and efficiently as possible. Through meta-reasoning an agent is able to explicitly consider goals before committing to them, and consider courses of action before executing plans. In this paper, we argue that although seldom considered, a flexible meta-level reasoning component is a valuable addition to any agent architecture. We describe such a component for use in BDI architectures, underpinned by a model of motivation and a motivation-based description language, and demonstrate its effectiveness empirically.

1 Introduction

In order to act effectively in any complex environment, autonomous agents must have control over their internal state and behaviour [1]. To exercise this control an agent needs some means of reasoning about its internal state, often in a process known as meta-level reasoning (or meta-reasoning). This is higher level reasoning about the reasoning process itself, and in agent systems it is commonly used in enforcing rationality in the choice of goals and actions performed by an agent, ensuring that an agent behaves as effectively and efficiently as possible. Through meta-reasoning an agent is able to explicitly consider goals before committing to them, and consider courses of action before executing plans, in opposition to simply reacting to events in the environment.

Indeed, several efforts towards refining the deliberation process, which can be viewed as meta-level reasoning, have been proposed recently. For example, meta-reasoning can be used to optimise task scheduling using some utility measure [2], and to improve and manage concurrent goal execution by exploiting opportunities and avoiding conflicts [3,4]. Most of these strategies rely on optimising agent behaviour by comparing some intrinsic characteristic of an agent's plans, execution time, or some abstract notion of utility. While improvements brought about by these techniques underline the benefits of meta-reasoning, most existing work does not treat meta-reasoning independently from the more common action-directed reasoning, and we believe that an agent architecture should have a separate such component.

For example, the widely known and used BDI architectures are usually of two types — procedural and declarative — and in both cases, there are advantages of meta-reasoning. Procedural architectures require detailed plans and triggers to be described by the designer, hence conflicts must be foreseen, with conflict resolution strategies embedded in the procedural plans. In procedural languages, specifying meta-reasoning separately from the plans removes the need to replicate internal *management* code

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 204–214, 2007.

throughout the plan library, facilitating development. Alternatively, declarative architectures are defined by desired states to be achieved and capabilities with which an agent can achieve them, where an interpreter selects capabilities to achieve goals, and conflict resolution must be done by this interpreter. In declarative languages, the lack of some goal selection policy means that goals and plans are selected arbitrarily, since in theory the designer does not specify precisely *how* goals are to be achieved.

In this paper we argue that models of *motivated* behaviour [5,6] can provide a valuable abstraction for the specification of meta-reasoning, specifically in the context of the BDI model. In particular, we describe how reasoning rules can be described using a motivational language, allowing the specification of customised meta-level behaviours. Integrating this with the AgentSpeak(L) architecture, we demonstrate empirically that such explicit meta-reasoning can bring significant improvements. The paper is organised as follows: Section 2 gives an overview of recent research on motivations; Section 3 describes an extended agent interpreter augmented with a motivation-based meta-level reasoner; Section 4 reports on the results obtained from experiments performed using our architecture; and, finally, Section 5 compares this work with existing efforts and outlines the main results and future work.

2 Motivations

Understood as the root cause of future-directed behaviour, motivation has been studied by researchers in a variety of areas, such as psychology [7], ethology [8] and philosophy [5]. A psychology-inspired definition of motivation considers it as representing an individual's orientation towards particular classes of goals [7], while a philosophical definition [5] encompasses the concept of varying motivational strength linked to an agent's urgency in relation to its associated goals. Based on multiple sources, Mele [5] posits that motivation is a trait present in animals capable of representing goals and means to goals, both of which may be influenced by motivation. That is, a motivation may influence both the adoption of a goal and the choice of the means to accomplish goals. Motivations vary in strength, dictating the choice of behaviours, so that intentional actions stem from underlying motivations.

From an ethological perspective, motivation is commonly associated with *drives* and *incentives* [8,9]. Simply, drives are internally generated states resulting from the violation of an animal's homeostasis, such as the deprivation of food or the excess of a given hormone. Incentives are externally generated stimuli that increase certain motivations, such as the presence of abundant food causing an animal to feed [8]. Motivations have also been described as giving rise to a continuum of appetitive behaviours (causing agent to need something) leading to consummatory ones (that satisfy this need). Thus some behaviours result in the build up of strength of certain motivations related to appetitive behaviour, and when a motivation has reached a high enough level, consummatory behaviours for the mitigation of this motivation are triggered.

The analysis of the motivational rewards of certain actions can also provide a mechanism to prevent certain undesirable behaviours from occurring simultaneously (also referred to as *lateral inhibition*) [10], as in trying to look at a watch while holding a mug of coffee with the same hand. More generally, if one assumes that the consequences of

any action can be measured as affecting motivations either positively or negatively, then such values can be used to determine which plans an agent can execute simultaneously without incurring detrimental interference among these plans.

The aspect of motivation most commonly sought to be captured by computational architectures is the continuous representation of priorities as a means to determine the *focus of attention* at any given time [11,12,6]. This is important as it allows an agent with limited resources to concentrate its efforts on achieving goals that are relevant to it at specific moments, and to adapt such a concentration of effort to the current reality. Contrasting with the traditional process of goal selection based solely on environmental state, real biological systems often generate different plans of action under the same environment. Here, motivations can be modelled as a mechanism associated with internal *cues* that trigger goal generation in parallel with external factors [9]. Such cues can be seen as trigger conditions that, when activated, cause an agent to consider the adoption of a set of associated goals. They differ from the simple logical preconditions traditionally used in that they result from the dynamics of motivation strength.

3 AgentSpeak-MPL

The BDI model has been the focus of agents research for a significant time [13], and is still ongoing. However, the first instances of complete BDI logics [14] assumed an agent able to foresee all of the future ramifications of its actions as part of the process of deciding which courses of action to take. This assumption was clearly too strong if computationally-bounded BDI architectures were to be constructed. Subsequent agent architectures were designed to avoid unbounded computations by selecting courses of action as a reaction to particular events in the environment under the assumption that they bring about the desired goal. This type of *reactive goal adoption* mechanism [15] is not compatible with a proactive stance, since it forces any future-directed behaviour to be bound to a stream of events for an agent to trigger its goal selection.

This is also true for AgentSpeak(L), in which plans matching events are adopted. For instance, whenever an agent believes that a given block is on a table, a procedure to remove the block may be invoked. This amounts to simple reaction rather than autonomous behaviour, since there is no consideration of any other goals that might have been pursued elsewhere at the time. Furthermore, this method of *behaviour selection* fails to describe the reasons for goal adoption in a declarative sense. The question here is whether the agent should *always* react to new events and start deliberation immediately even if it might be pursuing other more important goals. We believe that in performing meta-level reasoning, an agent can effectively consider its courses of actions without the risk of unbounded computation, and pay heed to any other more important goals. Moreover, by using motivations as a means to evaluate future goals, the assumption of omniscience required for evaluating competing goals is replaced with a preference relation, which is more realistic.

In order to demonstrate the utility of motivations as an abstraction for meta-reasoning, we have extended an AgentSpeak(L) interpreter with a motivation-based meta-level reasoner. This extended interpreter, AgentSpeak-MPL, takes a motivation specification in addition to a regular AgentSpeak(L) agent specification, and is used by the motivation module to update motivational intensity, generate goals and mitigate motivations.

Algorithm 1. mBDI control cycle.

```
1: loop
2:    perceive the environment and update the beliefs;
3:    for all motivation m do
4:       apply f_i to m to update i;
5:    end for
6:    for all motivation m do
7:       if i > t then
8:          apply f_g to m to generate new goals;
9:       end if
10:   end for
11:   select a plan for the most motivated of these new goals and adopt it as an intention;
12:   select the most motivationally valuable intention and perform the next step in its plan;
13:   on completion of an intention apply f_m to each motivation to reduce its intensity;
14: end loop
```

3.1 A Language of Motivation

An Abstract Motivation Model. In order to create the motivation component for our experiments, we adopt the model of Griffiths *et al.* [6] as a base. Here, a motivation is a tuple $< m, i, t, f_i, f_g, f_m >$, where m is the motivation name, i is its current intensity, t is a threshold, f_i is an intensity update function, f_g is a goal generation function, and f_m is a mitigation function. The model underpins the mBDI architecture [6], which in turn is based on the PRS/AgentSpeak architecture plus motivations. The reasoning cycle for an mBDI agent is illustrated in Algorithm 1. In brief, it consists of perceiving the environment and using this to update the intensity of each motivation, later generating goals for motivations with intensities that exceed their thresholds. When an agent adopts a new motivated goal, it selects plans to satisfy it, and then the most motivationally valuable intention for execution. When an intention finishes executing, the motivation is mitigated. This model was created for a procedural agent architecture, as is apparent from Steps 11 and 13 of the algorithm, which describes an intention as a plan to be executed, and mitigation of a motivation is equated to the completion of that plan.

So far we have described the abstract machinery that drives motivated control, and it is necessary to associate these abstractions to concrete motivations. Since different individuals can have different sets of motivations, they are affected by their motivations in varying ways, each with its own dynamics to allow evaluation of situations and achievement of goals according to an agent's unique priorities.

In order to allow a designer to describe motivational aspects for each agent, we therefore require a language to describe unique sets of motivations based on the abstract functions and data structures of the mBDI model. In consequence, we have designed a language centred on the three abstract functions of mBDI model: intensity update; goal generation; and mitigation. Concrete versions of these functions are essentially mappings between beliefs and an intensity value in the case of intensity update and mitigation, or new goals for the goal generation function. These functions are specified for each individual motivation, of which the agent can have several.

At a high level, each motivation is composed of an identifier, an intensity value, a threshold, and the name of a concrete function to be used for each of the required abstract functions of our motivation model, as follows:

$$\langle processBay, I, 10, f_i(Beliefs), f_g(Beliefs), f_m(Beliefs)\rangle$$

Whenever the intensity of a motivation reaches the declared *threshold* as a result of the intensity update function, it is said to be *activated*, and the goal generation function is invoked, after which the mitigation function is invoked to verify if the condition for the motivation to be mitigated is reached. Within the declaration of each concrete function, details of the mapping process are described, so with an intensity update function, the mapping consists of belief-value correspondences, since we are using new percepts to update a motivation intensity, while with a goal generation function, the mapping is a series of belief-goal associations, since this function aims to generate goals given the perceptions which activated a motivation. We consider each of these in detail below.

Intensity Update and Mitigation Functions. As we have seen, the functions for updating the intensity of, and mitigating, a motivation need to provide some kind of mapping between perceptual data and an intensity variation. As a result, our *language of motivation* allows the specification of a mapping between belief formulas and an arithmetic expression expressing how the intensity level should be modified as a result of the beliefs being true. Any specific mapping is represented as $log\_expr \rightarrow arithm\_expr$, where log_expr is a logical expression on the beliefs (*e.g.* a(X) & b(Y)), and $arithm\_expr$ is an arithmetic expression (*e.g.* X+2). An example of such a mapping is shown below:

$$f_i(Beliefs) = \begin{cases} over(P, bay1) \wedge batt(10) \rightarrow 2 \\ occupied(agent) \rightarrow -1 \end{cases}$$

Here, the intensity of the motivation to process a packet is increased by 2 whenever the agent believes a new packet has arrived in loading bay 1 (*i.e.* $bay1$) and it has a battery level of 10. It is important to notice that this language deals exclusively with beliefs, both intrinsic ones and those resulting from perception, whereas some motivation models assign values to actions and, by doing so, conform to a procedural view of reasoning. The mitigation function provides a mapping that is syntactically the same as the intensity update function but, according to our model of motivations, is only invoked when an intention is adopted to satisfy its associated motivation.

Goal Generation. Aside from mapping beliefs into perceptions, we must also describe the mapping of beliefs into goals. Since goal generation is only invoked when the motivation threshold is exceeded as a result of intensity accumulation, our language allows the specification of additional constraints before a goal is generated, or the unconditional generation of goals through the *true* condition. Similar to intensity update in that mappings start from a logical expression over beliefs, the target of this mapping are new goals to be achieved as a result of the intensity reaching the threshold in the motivation containing this goal generation function. This is illustrated below, where the agent generates an event to sort a packet located over $bay1$ whenever the goal generation function is invoked. Here, the constraint on the left of the mapping exists only to

Table 1. Example of a set of motivations

$\langle processBay1, I, 10, f_i(Beliefs), f_g(Beliefs), f_m(Beliefs)\rangle$, where

$$f_i(Beliefs) = \begin{cases} over(P, bay1) \wedge batt(10) \rightarrow 2 \\ occupied(agent) \rightarrow -1 \end{cases}$$

$$f_g(Beliefs) = \{over(Packet, bay1) \rightarrow +!sort(Packet)$$

$$f_m(Beliefs) = \{over(Packet, pigeonHoles) \rightarrow -20$$

allow the unification of the packet name with the goal to be generated.

$$f_g(Beliefs) = \{over(Packet, bay1) \rightarrow +!sort(Packet)$$

Example. A complete example of a motivation described using our language is shown in Table 1, specifying the motivation to process packets arriving from loading bay 1 from our previous examples. The motivational intensity starts to increase as soon as the agent detects an unsorted packet over $bay1$, until it reaches the the threshold of 10. Once the threshold is reached, the goal generation function adds a goal to sort this packet. Finally, the agent assumes the motivation is mitigated when it perceives the packet to be over the pigeonholes, diminishing the motivational intensity accordingly.

3.2 Integration with AgentSpeak

In our model, motivation intensity is a function of the perceived world state, so most of the motivation machinery is associated with the agent's belief update function. Each motivation data structure comprises an intensity value, a threshold value, and functions for intensity update, goal generation and mitigation. These data structures are updated as a result of the agent perception of the world, as illustrated in the activity diagram of Figure 1. When an agent receives new perceptions, it updates its belief base which is immediately inspected by the *intensity update function* associated with all motivations affecting this agent. During the update process, if the motivational intensity of any motivation reaches its threshold level, the *goal generation function* is invoked, also inspecting the belief base, and generating a set of goals based on the current world state. Finally, the belief base is inspected by the mitigation function to determine if any of the motivations triggered previously have been mitigated, in which case motivations are adjusted accordingly. In practice, the intensity update performed by the mitigation function is equivalent to that of the intensity update function. However, this update can only apply to motivations that had been previously *activated* as a result of their threshold levels being reached, and had generated goals.

4 Experiments and Results

In order to evaluate the potential improvements to agent efficiency, we adapted the scenario used by Duff *et al.* [16] to outline the advantage of proactive maintenance

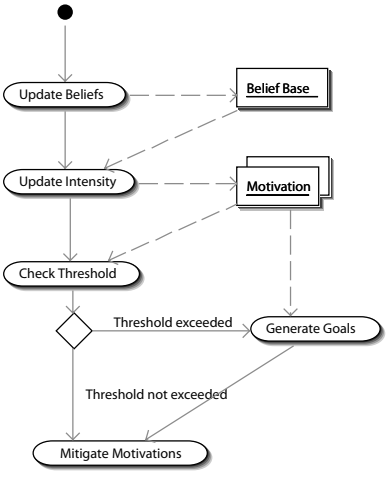


Fig. 1. Activity diagram of a motivated belief update

goals in agents. This scenario consists of a Mars rover capable of moving about a two-dimensional environment, in which movement consumes energy from the rover's batteries as it moves. Each unit of distance covered by the rover drains one unit of battery energy, and the rover can recharge at the mothership located at the centre of the environment. In the scenario, goals consist of waypoints through which the rover must pass in its exploratory expedition. A varying number of goals was given the agent to assess four test parameters: effective movement, consisting of the distance travelled towards waypoints; supply movement, consisting of the distance travelled towards the mother ship for recharging; wasted movement, consisting of the distance travelled to a waypoint that was wasted due to the need to recharge halfway through getting to a waypoint; and the number of intentions dropped to avoid complete battery discharge.

In this context, a more effective agent travels the least wasted distance, as well as the least supply distance (optimising battery use). Regarding the reasoning process itself, a more rational agent can be seen as one that drops the least amount of goals, since reasoning on adopting and managing ultimately dropped goals is also wasteful.

Our experiments consisted of submitting an increasingly larger number of waypoints, randomly generated for each set varying from 10 to 100 waypoints, to three different agents. The baseline of performance for the experiments was established by an agent with an infinite amount of energy, giving optimal movement to navigate through the entire set of waypoints, since there is no need to move to the mothership. These waypoints were also submitted to a traditional AgentSpeak(L) agent, which must monitor its battery level and recharge it before being too far from the mothership. Its strategy is to navigate to the waypoints in the order they arrive, without prior consideration of battery level. Whenever it is about to move to a position beyond reach of the mothership, it drops the goal to navigate the waypoints and adopts the goal to move to the mothership and recharge. A third agent used AgentSpeak(MPL) , driven by two motivations to navigate and to keep a safe battery level. These motivations behave as a sort of bilateral inhibition mechanism, in which a high intensity for the motivation to have a safe battery

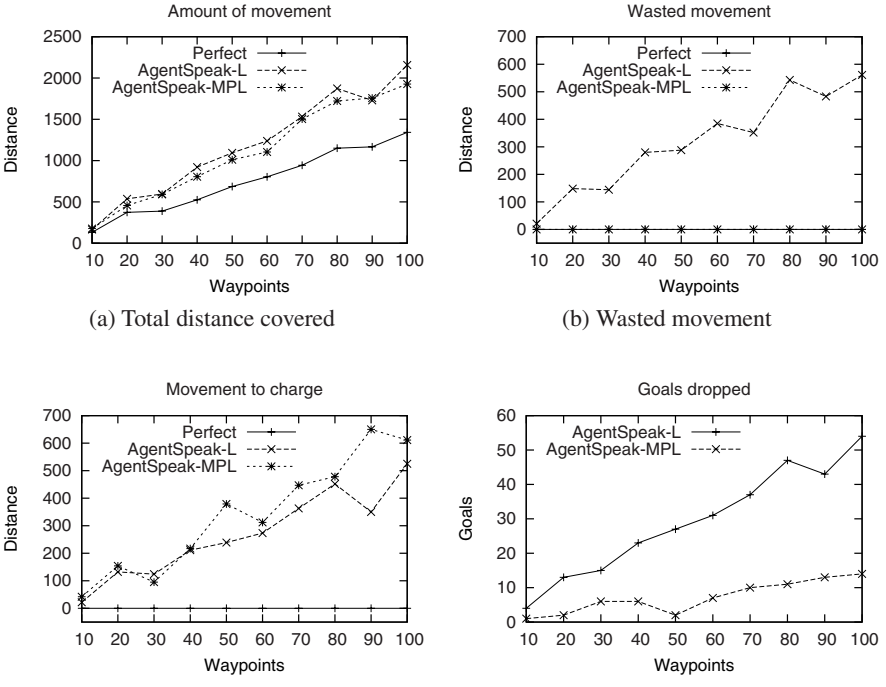
Table 2. Plan library size comparison

	AgentSpeak(L)	**AgentSpeak-MPL**
# atoms	119	95
# plans	19	10

level suppresses the intensity of the motivation to navigate. In more detail, whenever it perceives a new waypoint, the motivation to navigate is stimulated, and when it reaches its threshold, a goal to navigate to this waypoint is generated. The motivation to navigate, however, is suppressed if the battery level is not sufficient to reach that waypoint or if the charge spent doing so will place the agent in a position too far from the mothership. Conversely, the motivation to keep the battery level safe is stimulated by these two battery-related conditions. When the intensity of this motivation reaches its threshold, a goal to move to the mothership and recharge is generated. The result of this is that a goal to navigate to a waypoint should not be generated unless the rover has enough battery to move to it and then to the mothership.

Because the traditional AgentSpeak(L) agent starts executing plans to navigate as it perceives new waypoints, it only detects a critical battery level after having moved some way towards its target position, resulting in a waste of movement actions and a larger total distance covered. On the other hand, the effect of considering the amount of charge before starting to execute a plan to navigate is that no movement is wasted, and thus the total distance covered is smaller, as illustrated in Figure 2a, which compares the total distance covered by each of the agents. In these experiments, the motivated agent had to cover an average 6% less distance than the traditional AgentSpeak(L) agent. The traditional agent had to cover an average of 54% more distance than the baseline agent, compared to 45% for the motivated one, as illustrated in Figure 2b. Figure 2c illustrates the distance covered while moving towards the mothership for charging, where the motivated agent appears to move more than the traditional agent. This is a side-effect of the motivated agent always moving towards the mothership *intentionally*, rather than a side-effect of moving towards a waypoint closer to the mothership, which is corroborated by the smaller amount of total movement shown previously. The last evaluation parameter for this experiment relates to the number of goals dropped as a result of higher-priority goals being pursued. Goals to navigate must be dropped by the rover whenever it has to move back to the mother ship to recharge. Goals that are eventually dropped amount to wasteful deliberation, which rational agents should minimise. Here, the difference between the two approaches is more pronounced, with the motivated agent dropping an average of 75% fewer goals than the traditional agent, as shown in Figure 2d.

Specification size. In terms of the size of the agent specification, illustrated in Table 2, traditional AgentSpeak(L) uses a larger plan library to perform the meta-reasoning required to manage concurrent goals, and to allow the prioritisation of the goal to recharge. On the other hand, the motivated agent's plan library, which was derived from the former, requires a significantly smaller number of plans, since the motivation module ensures that goals are generated once per motivation until being mitigated, while also taking care of bilateral coordination.



(a) Total distance covered

(b) Wasted movement

(c) Movement towards the mothership

(d) Number of dropped goals

Fig. 2. Graphics for the rover experiment

5 Related Work and Conclusions

In this paper we have described an extended agent architecture that includes an explicit meta-reasoning module with an accompanying motivation-based specification language. This follows other recent efforts in adding such reasoning to agent architectures: Raja and Lesser [2] use meta-reasoning to allow the scheduling of existing tasks based on a pre-defined preference relation; Pokahr *et al.* [4] and Thangarajah *et al.* [3], rely on a technique that summarises the effects of plans considered for adoption and analyse positive and negative interactions to avoid conflicts and maximise opportunities; and, finally, Shaw and Bordini [17] use a Petri-Net representation of plans to reason about their utility and potential conflicts. These efforts improve agent efficiency by focusing on specific areas of the reasoning process to optimise. However, such optimisations rely on detailed knowledge of their underlying architectures [4,3], on particular plan representations [17], or on some abstract notion of utility to allow prioritisation [2], and all use a single, static strategy to improve agent efficiency. Our approach differs in that it enables the flexible specification of meta-level behaviour, which, in theory, could be used to specify optimisations similar to all of these efforts.

Given the advantages of having meta-reasoning capabilities in an agent architecture, we have described how motivations can be used as an abstraction mechanism to define meta-level behaviour. We have created a module of motivations with an associated language based on existing work from the computational motivation literature, and have

used it in a prototype based on an extended AgentSpeak interpreter. Our experiments have shown that our model can achieve the same kind of improvement that other reasoning optimisation strategies have achieved, while specifying meta-level behaviour separately from action-directed behaviour also results in a simpler agent description. Even though we used AgentSpeak(L) as our test platform, this approach can be easily employed for other BDI-based agent languages as well. In the future, we intend to perform more experiments using our language and motivation model to determine potential improvements for richer meta-reasoning capabilities.

Acknowledgments. The first author is supported by *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES) of the Brazilian Ministry of Education. Thanks to Andrew Jones for useful discussions regarding the content of this paper.

References

1. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence 117(2), 277–296 (2000)
2. Raja, A., Lesser, V.: Meta-level reasoning in deliberative agents. In: Proc. of the IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, pp. 141–147. ACM Press, New York (2004)
3. Thangarajah, J., Padgham, L., Winikoff, M.: Detecting & exploiting positive goal interaction in intelligent agents. In: Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 401–408 (2003)
4. Pokahr, A., Braubach, L., Lamersdorf, W.: A goal deliberation strategy for BDI agent systems. In: Proc. of the 3rd German Conf. on Multiagent System Technologies, pp. 82–93 (2005)
5. Mele, A.R.: Motivation and Agency. Oxford University Press, Oxford (2003)
6. Griffiths, N., Luck, M.: Coalition formation through motivation and trust. In: Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 17–24 (2003)
7. Morignot, P., Hayes-Roth, B.: Motivated agents. Technical report, Stanford University (1996)
8. Balkenius, C.: The roots of motivation. In: Proc. of the 2nd Int. Conf. on Simulation of Adaptive Behavior, pp. 513–523 (1993)
9. Munroe, S.J., Luck, M., d'Inverno, M.: Towards motivation-based decisions for worth goals. In: Proc. of the 3rd Int. Central and European Conf. on Multi-Agent Systems, pp. 17–28 (2003)
10. Grand, S., Cliff, D.: Creatures: Entertainment software agents with artificial life. Autonomous Agents and Multi-Agent Systems 1(1), 39–57 (1998)
11. Norman, T.J., Long, D.: Alarms: An implementation of motivated agency. In: Tambe, M., Müller, J., Wooldridge, M.J. (eds.) Intelligent Agents II - Agent Theories, Architectures, and Languages. LNCS, vol. 1037, pp. 219–234. Springer, Heidelberg (1996)
12. Cañamero, D.: Modeling motivations and emotions as a basis for intelligent behavior. In: Proc. of the 1st Int. Conf. on Autonomous Agents, pp. 148–155 (1997)
13. Georgeff, M., Pell, B., Pollack, M.E., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In: Rao, A.S., Singh, M.P., Müller, J.P. (eds.) ATAL 1998. LNCS (LNAI), vol. 1555, pp. 1–10. Springer, Heidelberg (1999)
14. Rao, A.S., Georgeff, M.P.: Formal models and decision procedures for multi-agent systems. Technical Report 61, Australian Artificial Intelligence Institute (1995)

15. Norman, T.J., Long, D.: Goal creation in motivated agents. In: Wooldridge, M.J., Jennings, N.R. (eds.) Intelligent Agents. LNCS, vol. 890, pp. 277–290. Springer, Heidelberg (1995)
16. Duff, S., Harland, J., Thangarajah, J.: On proactivity and maintenance goals. In: Proc. of the 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 1033–1040 (2006)
17. Shaw, P.H., Bordini, R.H.: Towards alternative approaches to reasoning about goals. In: Proc. of the 5th Workshop on Declarative Agent Languages (2007)

On Complex Networks in Software: How Agent–Orientation Effects Software Structures

Jan Sudeikat[1] and Wolfgang Renz

Multimedia Systems Laboratory,
Department of Information and Electrical Engineering,
Faculty of Engineering and Computer Science,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
Tel. +49-40-42875-8304
{wr,sudeikat}@informatik.haw-hamburg.de

Abstract. Software-Engineering provides techniques to ease handling the essential complexity of software. A number of engineering paradigms and architectures have been devised and each generation claims to relieve future development efforts. But to date little is known about how different development approaches affect the underlying implemenation structures, making their contributions arguable. Recently, the statistical analysis of large–scale modular software systems – represented as directed graphs – revealed complex system characteristics, namely *scale–free* and *small–world* phenomena. In this paper, we argue that the exhibited network characteristics reflect utilized design approaches and apply graph analysis to examine the structural differences imposed by the utilization of Agent–Oriented Software Engineering. As this novel development paradigm proposes autonomous and pro–active entities as an atomic design and development metaphor for complicated and inherently distributed software systems, an initial analysis and comparison of graphs abstracting both agent– and object–oriented system designs reveals structural differences which suggest that agent autonomy influences the resulting underlying implementation structures.[1]

1 Introduction

Building complicated software structures is a complex task. The *essential complexity of software* [1] is a consequence of both (1) the high number and flexibility of constituent parts as well as (2) the complex interactions among them. Engineering paradigms and architectures are constantly revised an refined in order to relieve development processes and facilitate the management of the growing

[1] Jan Sudeikat is doctoral candidate at the Distributed Systems and Information Systems (VSIS) group, Computer Science Department, University of Hamburg, Vogt–Kölln–Str. 30, 22527 Hamburg, Germany.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 215–224, 2007.

complexity of future applications. While engineering paradigms provide different design metaphors, implementation tools/frameworks commonly rely on object–oriented implementation languages, providing a common ground for comparison and evaluation. Yet little is known about the relation of development procedures and the resulting structures in large–scale software.

A prominent novel development approach is *Agent–Oriented Software Engineering* (AOSE), which proposes *autonomous* and *pro–active* entities that sense and effect their environment – so–called *agents* [2,3] – as a design metaphor for inherently distributed systems [4]. Agent–oriented development efforts utilize dedicated programming languages and middleware platforms [5], which allow individual agent implementations to be executed autonomously.An established approach to agent design are intenional agents that pro–actively pursue goals prescribed according to platform dependent formalizations [6]. Implementation frameworks are based on object–oriented programming languages, leading to executables in these languages. While the benefits of this design metaphor has been argued [4], the evaluation of the benefits and drawbacks of this novel design metaphor has yet found minor attention.

Numerous systems – observable in nature and engineered – can be represented as graphs composed of *vertices* (*nodes*) connected via *edges*. Statistical analysis of these networks revealed insights in underlying system structures, particularly *small world* and *scale–free* phenomena attracted attention in literature [7]. *Small world* denotes that most vertex pairs are connected via *relatively short* path w. r. t. the overall network size. *Scale–free* means that the distribution of the connection degree of vertices is dominated by a *power law*. Recently, it has been found that modular software systems can be analysed in similar ways. Corresponding analysis of object–oriented software applications revealed that complex system phenomena are present in todays applications [8,9,10]. While these results can be used to examine system robustness, i. e. bug propagation [11], it is an open question how these characteristics raise from development efforts and relate to application domains as well as design approaches [12].

In order to approach the examination of these relations, we justify and apply statistical graph analysis to compare the structures that result from the application of different engineering paradigms. In the absence of clear–cut metrics for software architectures, this analysis approach is suitable to examine how design paradigms and metaphors influence applications structure, allowing reasoning of their benefits in application development. We compare the statistical properties of *agent*–based and *object*–oriented implementations and discuss how the inferred graph structures reflect design approaches. I. e. we present an indication that agent autonomy impedes the propagation of code changes.

This paper is structured as follows. In the next section, we present agent–orientation as an engineering metaphor and introduce the examined agent architecture and platform. The following section 3 discusses how software architectures can be mapped to graphs and the insights that can be obtained from this abstraction. In section 4 MAS case studies and object–oriented applications are examined. Finally, we conclude and give prospects for future work.

2 Applying Agent Oriented Software Engineering

Development of MAS is supported by dedicated agent–oriented programming languages [5] and specific types of middleware platforms, so–called *agent platforms*. These tools facilitate agent construction and execution following different agent architectures. Since sophisticated platforms are available, efforts to revise agent–oriented development methodologies attrac increasing attention [13]. These methodologies typically focus on concepts provided by target agent platforms / architectures [14] and provide tailored development processes, notations and modeling tools.

The *Belief Desire Intention* (BDI) architecture, is an established approach to the development of rational agents and among the few agent architectures that were adopted in industry [15]. Based on a theory of human practical reasoning [16], that describes rational behavior by the notions *belief*, *desire* and *intention*, facilitates the BDI architecture the construciton of intelligent agents by the core concepts of *beliefs goals* and *plans*, leading to a formal theory and an executable model [17].

Beliefs represent the local information of agents about both the environment and its internal state. The structure of the beliefs defines a domain–dependent abstraction of the actual environment. Agent goals represent individual desires, which are commonly expressed by target states to be achieved by agent actions. This general concept enables pro–active agent behaviors. Finally, plans are the executable means to agents. Agents can access a library of plans and deliberate which plans to execute, in order to reach intended goals. Single plans are not just a sequence of basic actions, but may also dispatch (sub–)goals.

The *Jadex* reasoning engine[2] [18] provides an execution environment for BDI-style agents on top of arbitrary distributed systems middleware. The individual agents consist of two parts. First, each agent is described by a so–called *Agent Description File* (ADF), which denotes the structure of beliefs, goals and plans and further structural information in XML syntax. Secondly, the activities an agent can perform are coded in plans, which are ordinary Java classes. Plan descriptions in the ADF reference the compiled Java classes and denote the conditions, events, or goals, which define the applicability of plans for certain execution contexts.

3 Graph Structures in Software Systems

Numerous systems – both natural and engineered – can be understood as graph structures, composed of *vertices* (nodes) and *edges* between them. Examples include *food webs*, *airline routes* and *coauthorships* (see [7] for a comprehensive review). Structural similarities in large–scale graphs were revealed by mathematical studies, typically focusing on *path length distributions*, *degree distributions* and *network resilience*. Graph representations of these systems may distinguish different types of vertices and edges or refine edges by annotating directions and

[2] freely available at: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

weight functions. In the following we focus on directed, possibly cyclic graphs (*digraphs*) composed of one node type connected via one type of directed edges.

Examinations of path length measure the mean *geodesic* (i. e., shortest) distance between vertex pairs in terms of numbers of edges. In numerous real world networks it has been found that the mean node distance is very small compared to the overall network size, the so–called *small–world* effect [19]. This implies that nodes can contact each other via comparatively few edge transitions.

The *degree* of a vertex is the number of edges connected to it. Histograms of the fractions (p_k) of vertices with a certain degree (k) are typically examined. In directed graphs *incoming* and *outgoing* vertex degrees need to be distinguished. While random graphs exhibit binomial or poison distributions, real–world networks are typically right–skewed, i. e. small degree values are far beyond the mean. These distributions often follow either *power law* $p_k \sim k^{-\alpha}$ or *exponential* distributions $p_k \sim e^{-k/\kappa}$, where α and κ are constant exponents. Power Law distributions behave scale–free, i. e. they obey $p(bx) = g(b)p(x)$, where g is independent from x. This implies that rescaling solely depends on the ratio b. In the following we will use power law and scale–free synonymously.

Network resilience is closely related to degree distributions and describes the stability of graphs under the removal of vertices. Since the systems – represented by graphs – typically rely on (transient) connections between nodes to function properly, the average path length increases when vertices are gradually removed, finally leading to disconnected vertex pairs. In power law distributed systems, the number of vertices with small degree distributions is high, making the graph resistant to random edge removal. On the other hand these graphs are vulnerable to the removal of the few highly connected vertices.

Recently these properties have been studied in modular software systems [9,8] (recent approaches are reviewed in [20]). These efforts are based on the observation that software designs, particularly object–oriented ones can be represented in digraphs [9]. Figure 1 (left) exemplifies how the directed relations in UML[3] diagrams can be interpreted in graph structures. Elements, i. e. *packages*, *classes* and *interfaces* are coarse-grained to nodes and the the relations between them, i. e. *aggregations*, *compositions* and *inheritance* are mapped to directed edges.

This view on software systems is established for expressing software metrics and in reverse engineering efforts. High out–degrees of artifacts indicate a code reuse and large in–degrees indicate complex functionalities offered to other elements (see e. g. [21]). Several implementations, including C++ open source applications [9,20,8], Java JDK and applications [22,23,24,12] and Smalltalk systems [10] have been examined. Power laws and small–world properties have been identified in degree distributions, while the actual results and parameters vary due to different mappings applied for graph derivation from implementations. Their presence has been explained by optimization [25], i. e. refactoring [26] efforts in large scale applications [20].

The obtianed results have let to the examination of *bug dynamics* [11,27], transferring the notion of network resilience to software systems. In order to

[3] http://www.uml.org/

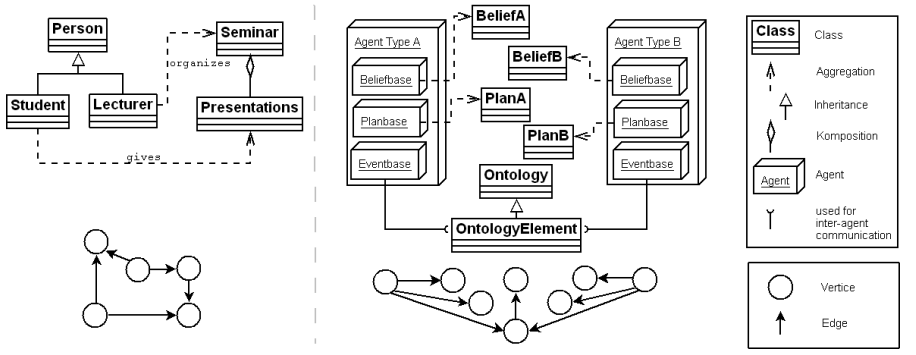


Fig. 1. Interpretations of object–oriented (left) and agent–oriented (right) implementations as plain graph structures. Objects and agents map to vertices and the relationships between them are abstracted to directed edges. The canonical view on a MAS (right) denotes that agent type declarations reference programming language classes, allowing a similar mapping.

anticipate the effects of bugs, i. e. node failures, their propagation can be calculated assuming that vertices connected to a faulty vertice will be negatively effected. In an *optimistic* scenario only nearest neighbors will be effected, in the *worst case* all vertices do not function properly. While software applications are build to be functional and design methodologies target modifiable, reusable and maintainable structures, the examination in [11] suggests that system stability can be inferred from the implementation structure. This structural stability is represented by the α and κ values of degree distributions, where lower values indicate more stable structures, due a limited propagation of code changes [11].

In the following, we examine how design paradigms influence the underlying implementation graph. Digraphs can be constructed for arbitrary MAS implementations by following how agent declarations relate to the finally executed implementation language. Figure 1 (right) gives a canonical view on a BDI–based MAS and shows how digraphs are constructed from MAS implementations. Agent declarations describe the agent logic and reference elements that provide the agent local knowledge (beliefs), executable means (e. g. plans) and semantic communication (ontology). On the implementation level these elements are conventional objects, consequently their relations can be mapped, similar to the purely object–oriented case (left).

4 Implementation Examination

In this section, we analyse the graph structures exhibited by object–oriented and MAS implementations. Our examination utilized the *Structural Analysis for Java*[4] (SAfJ) desktop application, which assists the analysis Java[5] applications

[4] freely available at: http://www.alphaworks.ibm.com/tech/sa4j
[5] http://java.sun.com

Table 1. Summary of the studied implementation networks

	Applications		
	Jadex (V 0.951 beta)	Ant (V 1.6.5)	Tomcat (V 6.0.0)
# Elements	1393	1574	1488
# Relationships	13532	12525	15503
Max. Dependencies	113	134	247
Average Dependencies	9.71	7.95	10.41
	Multi–Agent Systems		
MAS	Student Project	Marsworld	Cleanerworld
# Elements	259	34	81
# Relationships	1131	146	419
Max. Dependencies	55	17	25
Average Dependencies	4.37	4.29	5.17

by interpreting *extends*, *implements*, *uses* (method argument or returned from the method), and *contains* relationships which are present between *classes*, *interfaces*, and *packages*. SAfJ visualizes the exhibited network structure, calculates metrics, searches for undesirable node relationships (so–called *anti–pattern*) and allows to examine the effects of class changes. In the following four kinds of degree distributions are examined:

- Outgoing edges (dependencies) - # of dependencies to other vertices.
- Incoming edges (dependents) - # of vertices, dependent on a specific vertice.
- Affected - # of times a given object is affected when any other object changes.
- Affects - # of objects affected when a given object changes.

While the first two distributions measure in and out degree distributions (cf. section 3), the latter two distributions measure how code changes propagate through the implementation network. References to Java API classes (java.*, javax.*, com.sun.*), exceptions, inner classes and test classes were excluded from the examination. The examination of the MAS additionally ignores references to the agent platform API.

The here considered MASs are based on the Jadex platform (cf. section 2). Two example application (Marsworld, Cleanerworld) available with the Jadex distribution and a student project developed at Hamburg University[6] are examined. While the example scenarios exemplify agent development in simple game settings, the student project was created by 40 Students collaborating for 14 weeks in 7 teams to develop a MAS scenario, inspired by a case study discussed throughout [28]. Based on this agent–oriented design of a web–based book store, the individual teams were responsible to develop agent type implementations as well as a communication ontology and database schema. The resulting graph structure is related to the software graphs exhibited by the Java–based Jadex system implementation itself (cf. section 2) as well as the apache *Ant*[7] and *Tomcat*[8] open source systems. Table 1 summarizes the examined implementations.

[6] http://vsis-www.informatik.uni-hamburg.de/teaching/ss–05/rva
[7] http://ant.apache.org
[8] http://tomcat.apache.org

4.1 Graph Analysis

Examination of the *in*– and *out*–degree distributions revealed similar behaviors as reported in literature [20,12]. In order to accurately measure the tails of the degree distributions and reduce noise (common problems discussed in [7]), the measured (incoming) degree values are presented in log–log plots of the cumulative distribution function:

$$P_k = \sum_{k'=k}^{\infty} p_{k'} \tag{1}$$

The graph of the incoming degree (figure 2) can be fitted with a power law $\propto x^{-1}$ multiplied with a gaussian $e^{-\frac{x^2}{2r}}$ with r denoting the the number of relations. The Gaussian describes the size–dependent cut–off, where the number of relations scales with size squared. Figure 2 (right) denotes the distribution of affected elements, i. e. the number of times an element is affected when one element changes. These distributions follow power laws for small degree values, where the α value for the MAS implementation ($\alpha \approx -1.68$) is notably smaller than for the remaining object–oriented implementations (ranging from -1.08 to -1.28). Also in these plots the cut–off of the MAS implementations scales with the square–root of the number of relations, but it is very sharp–edged according to the small statistics available. According to the different exponents for the object–oriented systems the cut–off is shifted. The lower exponents and the sharp–edged cut–offs indicate that the number of classes that cause high sensitivity to code changes, i. e. bugs, is smaller in the MAS implementations.

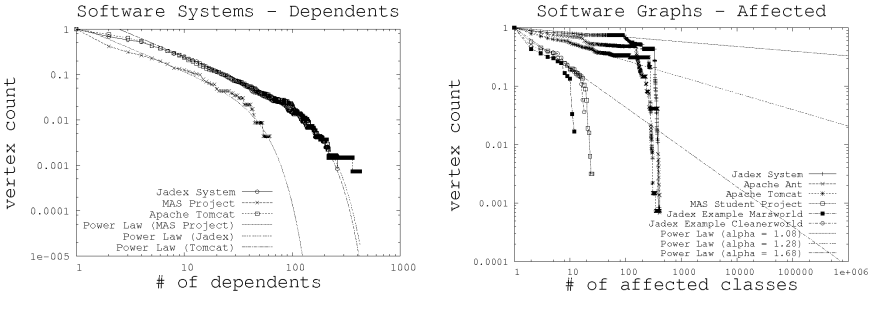


Fig. 2. Left: The cumulative distribution of the dependents (incoming degree) of individual nodes. Degree distributions follow power laws. Right: The cumulative distribution of affected nodes.

4.2 Discussion

In [12] different distributions have been observed across systems in the same implementation language, raising the question if these differences are caused by the demands of different application domains or system designs. We follow this direction and observe a qualitative difference between system designs following

two design paradigms. Our examination of degree distributions confirms power law observations in software systems (cf. section 3). However, the examination of the affected distribution suggests a structural difference to purely object–oriented designed systems.

The plot of the affected distributions in figure 2 visualizes the (scaled) range of elements that are sensitive to code changes. For the examined MAS implementations the exponents that controll this range are notably smaller, indicating less sensitivity throughout the systems. We speculate that this difference is caused by the inherent autonomy of agent implementations. Agents are designed to function autonomously and have been implemented independently from each other. Their typical sources of common references are the platform API (excluded from examination) and ontology objects utilized for agent communication. It is to note that this increase of insensitivity does not come for free. Agent declarations convey domain knowledge to enable pro–active *reasoning* about individual agent actions and the validation of agent conclusions is challenging [29]. While the sharp bend in the distribution may indicate typical application or platform dependent building blocks, i. e. agent size, this indication needs further evaluation.

Our examination confirms that power law distributions seem to be inherent to software systems and it is an open research question which distribution characteristics are desirable in software and how these are enforced by development procedures. Power law distributed systems are vulnerable to the removal of the highly connected nodes [7]. In modular software systems, it is typically intended to compose systems of merely independent and interchangeable components (*Lego Hypothesis* [12]), leading to the absence of system–wide important and therefore highly connected nodes. Self–similar distributions contradict the idea that atomic building blocks of a larger size than 1 are present [12]. The examination of bug propagation in [11] reveals another important property of power law distributed systems. Their relatively short average path length supports bug (and respectively code changes) to propagate through the graph.

While degree distributions in software structures seem to be inherently power law dominated, our comparison of the affected distributions suggests that an aim of software engineering procedures may be the raise of distribution exponents and/or to enforce cut–offs, consequently impeding change / bug propagations. The resulting stability increase in MASs is observable in the affected distribution, i. e. by their ability to quickly leave the power law and follow exponential distributions (figure 2). It remains to be examined how agent and MAS architectures contribute to these structural properties. The here analysed MAS suggest that agent communication ontology classes provide small sets of highly referenced nodes while cognitive agent architectures seem to facilitate partitioned structures. These indications and their impact require further examination.

5 Conclusions

In this paper statistical graph analysis has been applied to examine the implementation code structues that originate from two different design metaphors. Namely,

we compared agent–oriented applications to freely available open source systems. The examination of vertex degree distributions confirms the presence of power law distributions, as reported in literature (cf. section 3), for both design approaches. In addition, the propagation of code changes, e. g. bugs and failures has been examined and a qualitative different behavior has been observed. It appeared that the MAS–based systems are more stable with respect to code changes.

These initial observations indicate that the utilized design metaphor influences the underlying implementation structure. This indication requires further evaluation. It remains to be examined if these structures are indeed inherent to MAS and how they can serve as a reliable measure for system stability. Future work will facilitate this by revising dedicated tool support for the examination how agent architectures contribute to implementation structures. In order to derive quality measures and metrics for general software systems, predictive models are necessary to explain how the identified distributions raise from development efforts. While this work focused on the differences between agent–oriented and object–oriented designs, it is also of interest to confirm the results for other development paradigms, leading to object–oriented implementations.

Acknowledgments

One of us (J.S.) would like to thank the *Distributed Systems and Information Systems* (VSIS) group at Hamburg University, particularly Winfried Lamersdorf, Lars Braubach and Alexander Pokahr for inspiring discussion and suggestions.

References

1. Brooks, F.P.: No silver bullet: essence and accidents of software engineering. Computer 20(4), 10–19 (1987)
2. Franklin, S., Graesser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) Intelligent Agents III. Agent Theories, Architectures, and Languages. LNCS, vol. 1193, Springer, Heidelberg (1997)
3. Odell, J.: Objects and agents compared. Journal of Object Technology 1(1) (2002)
4. Jennings, N.R.: Building complex, distributed systems: the case for an agent-based approach. Comms. of the ACM 44(4), 35–41 (2001)
5. Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A.E.F., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica 30, 33–44 (2006)
6. Braubach, L., Pokahr, A., Lamersdorf, W., Moldt, D.: Goal representation for BDI agent systems. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Programming Multi-Agent Systems. LNCS (LNAI), vol. 3346, Springer, Heidelberg (2005)
7. Newman, M.: The structure and function of complex networks. M. E. J. Newman. The structure and function of complex networks. SIAM Review 45(2), 167–256 (2003)
8. de Moura, A.P.S., Lai, Y.C., Motter, A.E.: Signatures of small-world and scale-free properties in large computer programs. Physical Review E 68 (2003)

9. Valverde, S., Sole, R.V.: Hierarchical Small Worlds in Software Architecture. ArXiv Condensed Matter e-prints (July 2003)
10. Marchesi, M., Pinna, S., Serra, N., Tuveri, S.: Power laws in smalltalk. In: ESUG Conference 2004 Research Track (2004)
11. Challet, D., Lombardoni, A.: Bug propagation and debugging in asymmetric software structures. CoRR cond-mat/0306509 (2003)
12. Baxter, G., Frean, M., Noble, J., Rickerby, M., Smith, H., Visser, M., Melton, H., Tempero, E.: Understanding the shape of java software. In: OOPSLA '06. Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming languages, systems, and applications, pp. 397–412. ACM Press, New York (2006)
13. Henderson-Sellers, B., Giorgini, P. (eds.): Agent-oriented Methodologies. Idea Group Publishing, USA (2005)
14. Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W.: Evaluation of agent–oriented software methodologies - examination of the gap between modeling and platform. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 126–141. Springer, Heidelberg (2005)
15. Benfield, S.S., Hendrickson, J., Galanti, D.: Making a strong business case for multiagent technology. In: Proc- of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (keynote paper, 2006)
16. Bratman, M.: Intentions, Plans, and Practical Reason. Harvard Univ. Press, Cambridge (1987)
17. Rao, A.S., Georgeff, M.P.: BDI-agents: from theory to practice. In: Proceedings of the First Int. Conference on Multiagent Systems (1995)
18. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A BDI Agent System Combining Middleware and Reasoning. In: Software Agent-Based Applications, Platforms and Development Kits, Birkhäuser (2005)
19. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature 393, 440–442 (1998)
20. Myers, C.R.: Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. Phys. Rev. E 68 (2003)
21. Henderson-Sellers, B.: Object–Oriented Metrics: Measures of Complexity. Prentice-Hall, Englewood Cliffs (1996)
22. Potanin, A.: The fox – a tool for java object graph analysis. Technical report, School of Mathematical and Computing Sciences, Victoria University (2002)
23. Potanin, A., Noble, J., Frean, M., Biddle, R.: Scale-free geometry in oo programs. Commun. ACM 48(5), 99–103 (2005)
24. Wheeldon, R., Counsell, S.: Power law distributions in class relationships. In: Third IEEE International Workshop on Source Code Analysis and Manipulation, vol. 0, IEEE Computer Society Press, Los Alamitos (2003)
25. Valverde, S., Cancho, R.F., Sol, R.V.: Scale-free networks from optimal design. Europhys. Lett. 60(4), 512–517 (2002)
26. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, Reading (1999)
27. Challet, D., Le Du, Y.: Closed source versus open source in a model of software bug dynamics. ArXiv Condensed Matter e-prints (June 2003)
28. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide. John Wiley and Sons, Chichester (2004)
29. Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W., Renz, W.: Validation of bdi agents. In: Programming Multi-Agent Systems. 4th International Workshop, ProMAS 2006. LNCS (LNAI), vol. 4411, pp. 185–200. Springer, Heidelberg (2006)

Simulating a Human Cooperative Problem Solving

Alexandre Pauchet[1], Amal El Fallah Seghrouchni[2], and Nathalie Chaignaud[3]

[1] LITIS - EA 4051
pauchet@insa-rouen.fr
[2] LIP6 - UMR CNRS 7606
Amal.ElFallah@lip6.fr
[3] LITIS - EA 4051
chaignaud@insa-rouen.fr

Abstract. We are interested in understanding and simulating how humans elaborate plans in situations where knowledge is incomplete and how they interact to obtain missing information. Our human interaction model is based on the speech act theory to model the utterances and it uses timed automata to describe the dynamics of the dialogs. Our human planning model is implemented as a hierarchical blackboard architecture which manages opportunistic planning. The system BDIggy we propose, is a concurrent implementation of the planning model and the interaction model through the BDI concept. This system is used to simulate the human processes during cooperative problem solving.

1 Introduction

Nowadays, heterogeneous multi-agent systems, compound of human agents and software agents, become commonplace. Therefore, humans frequently interact with intelligent systems deployed in open environments. When users encounter incompletely described situations, they have to acquire missing information with the system and reason about it. They often have to anticipate their problem-solving and build plans dynamically. We are convinced that studying human activities of problem-solving with incomplete information is essential to the design of agents which interact with humans. More precisely, it is necessary to integrate an explicit representation of the user's beliefs, abilities and intentions. This user's representation cannot be constructed without an accurate understanding of how subjects elaborate plans and how they interact to obtain missing information. The goal of this research is to develop a computational model of human interaction and of human planning, that may help to improve the design of software agents involved in heterogeneous multi-agent systems.

A psychological experiment was conducted, during which subjects have had to solve a problem with incomplete information. The problem submitted to subjects is related to a travel-agency application. Three salesmen are each in charge of a particular means of transport: the first one manages airlines, the second one manages railways and the last one manages taxis and coaches. Each of

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 225–235, 2007.

them has to organize a journey for his own client. This problem consists in satisfying several constraints: a starting city and an arrival city in France, a time of departure, a time of arrival, a number of travelers and a budget. None of the journeys can be undertaken using a single means of transport. Thus, the subjects have to cooperate to solve the 3 problems. To solve their problem, the subjects use a software interface which records the performed actions and the emails while an experimenter notes the verbal utterances of the subjects. Each solving produces three experimental protocols (recorded actions, emails and concomitant verbalization), one for each subject.

The experiment was carried out with 14 groups of 3 students who solved the problems. We analyzed 9 groups in order to design the cognitive models and 5 groups were used for the validation of these models. The 9×3 experimental protocols have been analyzed from two points of view: human planning and human interaction. We modeled the human cooperative planning in an opportunistic way, while our human interaction model is based on the speech act theory to represent the utterances and uses timed automata to describe the dynamics of the dialogs. The system BDIGGY we propose is a concurrent implementation of the planning and interaction models through the BDI concept. The validation of the models and the BDIGGY architecture is done by comparing the output of this system with the human protocols collected during the psychological experiment.

2 The Human Planning Model

Each protocol has been analyzed and the resulting planning model uses the notions of phases, states of mind, strategies, tactics, observations and personality.

2.1 The Phases

Each protocol can be divided into several phases which differentiate the situations of the problem-solving.

- Subjects build plans during *planning*.
- Minor obstacles are removed during *correction*.
- *Deadlock-solving* corresponds to re-planning.
- When the solution is complete, the subject *tests* it.
- During *checking*, subjects verify their plans.
- *Cooperation* is used to help another subject.

2.2 The States of Mind

The decision to perform an action depends on the attention paid to the different constraints (or *criteria*) of the problem. The subjects take a subset of criteria into account, directly linked with the constraints. This subset, called state of mind, evolves during the problem-solving process according to the situation. The criteria are domain dependent. Five criteria are defined for the travel agency problem: *price*, *timetable*, *transfer*, *number of travelers* and *booking*.

2.3 The Strategies

Strategies model the way plans are built. The model differentiates the *approach strategies* (type of planning the subjects use), and the *planning strategies* (order in which subjects build plans). We observed:

- 2 approach strategies: *sequential planning* and *parallel planning*.
- 4 planning strategies: *prospective*, *retrospective*, *centrifugal* (from the middle of the travel to the edges) and *centripetal* (from the edges to the middle).

2.4 The Tactics

Subjects can instantiate one planning strategy by several actions and objects. Tactics are used to model the different choices of actions to be performed. The tactics are domain dependent. Tactics are used to choose the different stages of the journey (via the largest towns, the most direct, *etc.*) or the means of transport (the cheapest, the fastest, *etc.*).

2.5 The Observations

During the solving process, any change in one of the ingredients (phase, state of mind, strategy or tactic) corresponds to a new episode. These changes are triggered by the observations that subjects can make about the current situation of the problem. When the verbalization is sufficient, the observations correspond to the verbal utterances of the subjects. 18 observations appeared (Ex: the journey is too expensive, a particular town has a railway station, the departure or arrival of a plane is not compatible with the current solution, *etc.*).

2.6 The Personality

The personality models the individual differences between subjects. Each experimental protocol describes the behavior of a subject and this general behavior depends on values of the features of the personality. Eight features are used:

- *thrifty*: how the subject is concerned with the stage prices,
- *opportunistic*: how often the subject uses parallel planning,
- *good estimator*: the subject's capability to estimate correctly a solution,
- *careful*: how few mistakes are made,
- *optimizer*: how the subject tries to minimize the travel,
- *precise*: how exhaustive are the stage information searched by the subject,
- *patient*: how long the subject waits before sending re-queries,
- *cooperative*: how the subject participate easily to the other's problem,

When analyzed, a personality (8 features with a value in $\{1, 2, 3\}$) was attributed to each protocol.

3 The Human Interaction Model

Each of the 9 groups of 3 protocols have been merged into a single file, respecting the temporality. Both the utterance and the discourse levels are considered.

3.1 The Utterance Level

At the utterance level, each message of the experimental protocols was analyzed individually. To refer to the speech act theory [1], the observed speech acts are either *descriptives*, *directives* or *commissives*. Messages were first matched with a performative from FIPA-ACL [2] or from KQML [3]. When no matching performative exist, a new one is proposed. Table 1 presents the performatives observed in the protocols with their frequencies. This list is exhaustive only concerning the information-search dialogs.

Table 1. The observed performatives

Descriptives:		
inform	S sends a piece of information to R. From FIPA-ACL.	*2.7%*
notUnderstood	S does not understand one of R's previous message. From FIPA-ACL.	*0.8%*
reply	S answers R. Inform-if/inform-ref in FIPA-ACL.	*39.6%*
thank	S thanks R. New performative.	*2.7%*
Directives:		
acceptProposal	S accepts an information proposal from R. From FIPA-ACL.	*0.9%*
cancel	S tells R not to take into account a previous message. From FIPA-ACL.	*1.6%*
query	S asks R for a piece of information. Query-if/query-ref in FIPA-ACL.	*43.0%*
refine	S asks R to detail one of R's previous query. New performative.	*5.4%*
refuseProposal	S refuses an information proposal from R. Reject-proposal in FIPA-ACL.	*0.3%*
Commissives:		
propose	S proposes to send information to R. From FIPA-ACL.	*2.6%*

A message is represented by the predicate

$$pMessage(A_S\ A_R\ P\ C)$$

with A_S the sender, A_R the receiver, P the performative and $C \in BEL \cup DES$ the content on which the performative P is applied. BEL is the set of beliefs and DES is the set of desires. $C = pB(\alpha) | pD(A\ \delta)$ where $pB(\alpha) \in BEL$ with α a predicate, and $pD(A\ \delta) \in DES$ with $A \in \{air, railway, road\}$ an agent and δ a predicate. A desire is attached to an agent. Frequently, $\alpha \in STA$ and $\delta \in STA$. STA is the set of stages described by the predicate

$$pStage(C_D\ C_A\ T_D\ T_A\ M\ N\ P\ R)$$

where C_D and C_A are the starting and the arrival cities, T_D and T_A are the departure and the arrival times, M is the means of transport, N is the number of travelers, P is the price and R indicates if the stage is booked or not.

A performative is applied to a mental state (a belief or a desire), the scope of which is a predicate. A performative and its content are closely linked as the illocutionary force and its proposition (noted $F(P)$ in the speech act theory):

- A *descriptive* is applied to a *belief*. It describes how the sender perceives the world.

- A *directive* is applied to a *desire of the sender*. The sender wants to receive an information and he sends this desire to the subject who has the information.
- A *commissive* is applied to a *desire of the receiver*. The sender supposes that the receiver has a certain desire.

3.2 The Discourse Level

Our discourse analysis is based on the work of Vanderveken [4] which extends the speech act theory to discourse. He splits conversations into *illocutionary acts*, introduces *mental states* as basic reasoning units and calls *exchange* a set of bounded messages. The experimental protocols were divided into exchanges, classified into 4 categories: *information queries*, *information proposals*, *spontaneous sendings*, *error processings*. Each of these exchanges is guided by the discourse goal of the initiator, according to the first performative he sent. Error processing exchanges are the only ones whose discourse goal differs from the performative type of the initial message. Their discourse goal is directive because the receiver of the erroneous message wants the sender to cancel it. The way exchanges are terminated defines their satisfaction. An exchange can be either *satisfied* (the interlocutor's goal happens) or not. The starting performative is used to classify the exchange and the ending performative, if it exists, defines the exchange satisfaction. An exchange can be considered as terminated by the interlocutors even without explicit emission of an ending performative. Spontaneous sendings of information only need one main act: the starting performative.

As messages are emails, time is important regarding to re-queries and exchange terminations. When a subject needs an information from another subject, if he still has not received any answer after a certain time, he will re-ask for information. Similarly, exchange without any explicit ending speech act is considered as terminated after a certain time. Timed automata [5] are used to model these exchanges and the temporality. 4 pairs of automata are designed to represent the observed exchanges, a pair of automata (an automaton for each interlocutor) for each type of exchange. Fig.1 describes the behavior of the initiator of an information query (Q_{ini}). Each state represents a particular situation during the exchange. A transition can be crossed when a message is sent or received, after a delay or if a variable reaches a particular value. The 4 pairs of timed automata have been tested on the whole experimental protocols to ensure they are exhaustive. For each automaton, the frequency observed in the experimental protocols of each transition is specified. Timed automata are used

- *to generate a message*: messages are produced following an automaton. When there is no determinism, the decision is made according to the current situation and the simulated agent's personality. If several transitions can be crossed, the decision is made randomly respecting the frequency of the transitions.
- *to interpret a message*: an automaton describes the expected messages. 2 interlocutors can manage several exchanges simultaneously so these expected events help to know if a message belongs to an exchange.

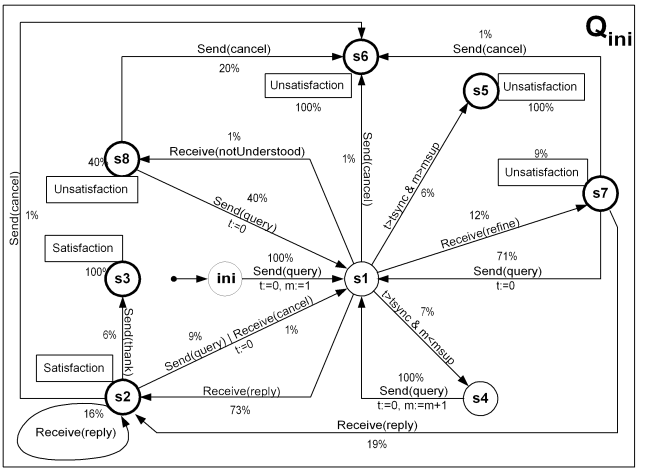


Fig. 1. Automaton of an information query

3.3 A Semantics of Performatives

This section describes the semantics associated with the sending/receipt of messages. The semantics of each performative is given by the generic reduction rule

$$[PreCond]\frac{A_X(si_1,...,si_n)\xrightarrow{send|receive(performative)}A_X(sf)}{a_1;...;a_n}$$

where *PreCond* are preconditions, A_X with $X \in \{ini, int\}$ is the automaton, $si_1...si_n$ are the states before processing the sending/receipt of a message, sf is the state after processing the sending/receipt of a message and $a_1,...,a_n$ are the internal "actions" to be performed in order to update the model.

We present here the semantics associated with the sending of a query in Q_{ini}.

Query

Syntax: *pMessage*(A_S A_R *query* *pD*(A_S S)) with A_S, $A_R \in \{air, railway, road\}$ and $S \in STA$.
Description: A_S can send a *query* if he wants the stage S, if S uses a different means of transport from A_S's one and if A_S has no belief about S.
Semantics:

$$\begin{bmatrix} pD(A_S,S) \\ \neg pMeans(S) \\ \neg pB(S) \\ \neg pB(\neg S) \end{bmatrix} \frac{Q_{ini}(ini,s2,s4,s7,s8)\xrightarrow{send(query)}Q_{ini}(s1)}{aUpdateTA(M)}$$

The postconditions are expressed with internal actions which can only affect the beliefs. Desires of the locutor and desires of the interlocutor are processed as knowledge (beliefs applied on a desire).

Moreover, when sending a message, the automaton states are updated with the action *aUpdateTA(M)*, *M∈MES*. This action checks if one of the opened automata matches the message, and modify it consequently. If there is no corresponding automaton, a new one is opened.

Using beliefs and desires in both the syntax and the semantics of a performative links the utterance and the discourse levels in our interaction model.

4 The Implementation

The human planning and interaction models are implemented into a new agent architecture, called BDIggy, which takes benefits from the BDI architectures. Further references to BDI model are based on dMARS [6]. A BDI agent includes a *queue of events* which stores internal and external events occurring in the system, some *beliefs*, a library of plans, a stack of *desires* and a stack of *intentions* (instantiated plans to reach the desires). The BDI interpreter runs as follows: first of all, events are updated generating new beliefs. Then, new desires are calculated matching plans of the library according to the beliefs. One of these plans is selected for execution and put into the intention stack. Finally, an intention is selected and its plan is performed, and so on.

A BDIGGY instance is an agent written in Java which simulates a subject and generates artificial protocols. It is worth emphasizing that the only information shared by the agents is passed through the messages. In fact, three agents run concurrently, forming a multi-agents system.

The BDIGGY architecture (see Fig. 2) includes the *perception module* which manages the agent's representation of the world. It analyzes the environment and generates beliefs. It contains a description of the problem (the agent's personality to simulate and the travel to be built), the domain knowledge and a representation of the current situation. All of them are beliefs and constitute the agent's *memory*.

The planning module is a hierarchical blackboard architecture, whose domain controllers manage the ingredients of the model. The module is initialized with a personality and a problem to be solved. From a description of the current situation, it generates the necessary observations and builds an episode (a phase, a state of mind, strategies and tactics), which can be considered as a short-term

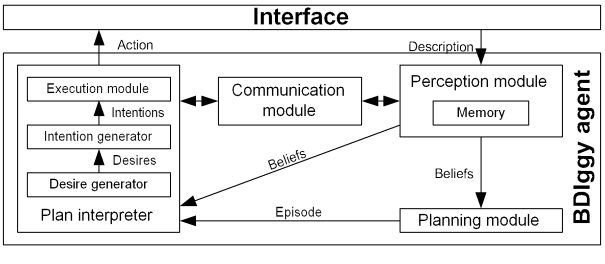


Fig. 2. The BDIGGY architecture

abstract plan. An episode leads the system to perform actions. When a series of actions is performed, new observations can be made that create a new episode. This incremental process enables to build plans in an opportunistic way. The planning module replaces the plan library of the BDI architecture, but plans are dynamically constructed.

The plan interpreter works as the BDI interpreter. The *desire generator* translates the abstract plan into several desires. The *intention generator* refines a desire into intentions. The *execution module* performs the actions of an intention. During parallel planning, the plan interpreter manages as many desire and intention stacks as there are parallel plans. Parallel plans correspond to different solutions for the problem, *i.e.* travels crossing different cities but with the same starting city and the same ending city. In the desire generator, a desire can be *abstract* or *elementary*, *instantiated* (refined into intentions) or *non-instantiated*, *satisfied* or *unsatisfied*, *successful* or *unsuccessful*, *active* or *suspended*. In the intention generator, an intention can be *active* or *suspended*.

The *communication module* implements the human interaction models. The perception module delegates the communication module to understand the received messages and the execution module delegates the communication module to write the messages. The communication module can add beliefs in the memory when receiving or sending messages. When a re-query has to be sent, according to the timed automata, a belief is also added to the memory which can possibly be processed by the plan interpreter during the next cycle.

5 Simulation and Validation

To simulate the solving processes of the travel-agency problem, 3 BDIGGY agents have been running simultaneously, generating new artificial protocols. The validation is done by hand analyzing a set of real and artificial protocols. The real protocols are drawn from the 5×3 protocols that had been put aside after the experiments. 10×3 protocols have been simulated, each having the personality of one of the real protocols analyzed. Our validation method is based on a Turing-like test. From the sets of protocols described above, we randomly drew one set with 2×3 real protocols and 4×3 artificial protocols and a second set with 3×3 real protocols and 3×3 artificial. Since our system does not support natural language processing, the messages from the real protocols were manually translated into our *pMessage* language.

"Experts" (fellow researchers who did not work on this study) were asked to hand analyze these sets of mixed protocols and to classify them according to their type (human or artificial). Table 2 presents the results of this validation. Expert 1 analyzed the two sets, Expert 2 analyzed only Set 1 and Experts 3 and 4 analyzed only Set 2. Each cell contains the proportion of well-classified protocols in each category. For example, the first cell indicates that Expert 1 classified correctly two protocols over the 3 human protocols of Set 1. The Total column specifies how many protocols each expert has classified correctly, among the number of protocols he analyzed (Expert 1 analyzed 12 protocols and classified correctly 6

Table 2. The validation results

	Set 1		Set 2		
	Human	BDIggy	Human	BDIggy	Total
Expert 1	2/3	1/3	1/2	2/4	6/12
Expert 2	1/3	1/3	-	-	2/6
Expert 3	-	-	0/2	2/4	2/6
Expert 4	-	-	2/2	2/4	4/6

of them and so on). Since the experts made around 50% of error, we can conclude that the experts were not able to reliably separate the two classes of protocols.

6 Related Work

This study deals with three issues: cognitive modeling, cooperative planning and interaction. To our knowledge, there are few works integrating all these aspects.

The TRAINS project [7] and TRIPS [8] are close to this work since they develop an intelligent planning assistant that interacts with humans. [7] proposes a model for collaborative agents which integrates an individual problem-solving model, a collaborative problem-solving model and an interaction model in natural language. The communication model works as in [9]: the goal is to recognize the interlocutor's plan to emit an appropriate answer. The notions used (*situations*, *objectives* and *recipes*) are similar to the BDI concept.

The COLLAGEN project [10] aims at developping an application-independent *collaboration manager* to facilitate collaborative interaction between software agents and users. COLLAGEN uses the SharedPlan theory of human collaborative dialogs [11], [12]. The decomposition of plans in recipes is similar to the one used in TRAINS. The utterances and actions to produce are selected according to how they contribute to the discourse purpose.

The TRAINS and COLLAGEN projects are concerned with collaborative interaction whereas we focus on cooperative interaction and planning. The main differences with our work is that we aim at simulating cooperative human planning and we proposed a cognitive model of opportunistic planning.

In comparison with KQML [3] and FIPA-ACL [2], our utterance model is better adapted to human dialogs. The list of performatives is improved and the link between the performative and its content is integrated. Many papers deal with performative semantics. [13] brings important results in the formalization of the Speech Act theory. [14] is oriented by the satisfaction of the speech acts. [15], used in FIPA-ACL, contains modal operators for describing the beliefs, desires, uncertain beliefs and persistent goals of the agents. Our semantics is based on the BDI concepts to use a single representation linking planning and interaction.

Concerning the communication protocols, FIPA-ACL cannot take into account the dynamics of human dialogs. The Dooley Graphs [16] contain information about the situation and the protagonists of a dialog, but do not consider the time dimension. We choose timed automata to represent duration of communicative actions and to synchronize interactions inside exchanges. To our knowledge, no other work uses this formalism to model interaction.

7 Conclusion

The strength of this work is to propose a complete study, from the collection of the experimental protocols to the implementation of the simulation system and its validation. The cognitive models are based on the analysis of the observed behaviors. From the planning point of view, the cooperative planning model allows to build opportunistic plans. From the interaction point of view, an utterance is represented by a performative applied to a mental state (a belief or a desire). Timed automata are a powerful formalism to introduce recursiveness and time management in the discourse representation. The performatives and the timed automata are linked thanks to a semantics using beliefs and desires. The planning and interaction models are integrated into the BDIGGY architecture and are represented in a homogeneous way (through BDI) in the same system.

Because cooperation is enforced during the 3 simultaneous solving processes, planning and interaction are interleaved. Each subject depends on the others to acquire missing information and has to interact with them. Most of the time, planning is the prime activity and interaction is only a tool to obtain information: planning manages interaction. But when new information is acquired not corresponding to the subject's expectations, the plan has to be changed opportunistically: interaction manages planning.

The human model of cooperative planning can be re-used for other problem solving. Only the domain specialists, domain dependent, have to be reimplemented to support another problem. The interaction model is exhaustive concerning the information-search dialogs. It has to be extended to other kind of dialogs, such as collaborative ones. Concerning the BDIGGY architecture, the problem description (predicates contained by the memory) and the interpretation of episodes into abstract plans are also dependent of the problem, whereas the process of the architecture is general.

The ultimate goal of this work was to propose a computational model which implements the cognitive model in order to simulate human processes during cooperative problem solving. In future work, these new knowledge would be used to design operational systems able to interact efficiently with human end-users.

References

1. Searle, J.: Speech Acts – An Essay in the Philosophy of Language. Cambridge University Press, Cambridge (1969)
2. FIPA: FIPA communicative act library specification. Technical report, Foundation for Intelligent Physical Agents (2002)
3. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an agent communication language. In: CIKM'94, pp. 456–463 (1994)
4. Vanderveken, D.: Illocutionary logic and discourse typology. Revue Internationale de philosophie 55(216), 243–255 (2001)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical computer science 126, 183–235 (1994)

6. d'Inverno, M., Kinny, D., Luck, M., Wooldridge, M.: A formal specification of dMARS. Technical note 72, Australian Artificial Intelligence Institute, Carlton, Victoria (1997)
7. Allen, J., Blaylock, N., Ferguson, G.: A problem solving model for collaborative agents. In: Falcone, R., Barber, S., Korba, L., Singh, M.P. (eds.) AAMAS 2002. LNCS (LNAI), vol. 2631, pp. 774–781. Springer, Heidelberg (2003)
8. Allen, J., Ferguson, G.: Human-machine collaborative planning. In: NASA Workshop on Planning and Scheduling for Space, Houston, USA (2002)
9. Traum, D.: A Computational Theory of Grounding in Natural Language Conversation. PhD thesis, University of Rochester (1994)
10. Rich, C., Sidner, C.: COLLAGEN: A collaboration manager for software interface agents. User Modelling and User-Adapted Interaction 8(3/4), 315–350 (1998)
11. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
12. Lochbaum, K.: A collaborative planning model of intentionnal structure. Computationnal linguistic 24(4), 525–572 (1998)
13. Cohen, P., Levesque, H.: Performatives in a rationally based speech acts theory. In: MACL'90, Pittsburgh, Pensylvania, pp. 79–88 (1990)
14. Singh, M.: Towards a formal theory of communication for multiagent systems. In: IJCAI'91, Sidney, Australia, pp. 69–74 (1991)
15. Sadek, M.: A study in the logic of intention. In: KR'92, Cambridge, pp. 462–473 (1992)
16. Parunak, H.V.D.: Visualizing agent conversations: using enhanced dooley graphs for agent design and analysis. In: ICMAS'96, Kyoto, Japan, pp. 275–282 (1996)

Supporting Agent Organizations

Estefania Argente, Javier Palanca, Gustavo Aranda, Vicente Julian,
Vicente Botti, Ana Garcia-Fornes, and Agustin Espinosa

Universidad Politecnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
{eargente,jpalanca,garanda,vinglada,
vbotti,agarcia,aespinos}@dsic.upv.es

Abstract. Organizational Theory concepts have been adapted to improve, extend and facilitate Agent Organizations modeling. Thus, three basic organizational topologies from which other more complex organizations can be composed have been identified. Moreover, this organizational modeling has been implemented in a specific MAS platform (SPADE) making use of Multiuser Conference technology. We have also defined several services for SPADE that ease the implementation of organization dynamics, such as member access and agent interaction control[1].

Keywords: Multi-Agent Systems, Organizations, SPADE.

1 Introduction

Organizational models have been recently used in agent theory for modeling coordination in open systems and to ensure social order in MAS applications [1]. Agent Organizations rely on the notion of openness and heterogeneity and include the integration of organizational and individual perspectives; and the dynamic adaptation of models to organizational and environmental changes [2].

In previous works [3,4] an analysis of MAS methodologies and agent platforms from the point of view of the organizational concepts was carried out. Most important examples of methodologies for agent organizations are OperA[5], OMNI [6] and MOISE+ [7]. Those methodologies describe organizations by means of its roles, interactions and social norms, relying on other agent methodologies for the design of the agent internals [2].

Regarding agent platforms, the most well-known agent platforms [3] offer basic functionalities to agents, such as AMS and DF services; but designers must implement nearly all organizational features by themselves, like communication constraints imposed by the organization topology. AMELI[8] platform helps designers to control obligations and norms of agents in an electronic institution, in which system structure is organized by means of interaction scenes and transitions. An scene is a communicative process between agents and there is not

[1] This work is partially supported by the TIN2006-14630-C03-01, TIN2005-03395 and GV06/315 projects of the Spanish government.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 236–245, 2007.

any predefined pattern for communication relationships inside a scene. JACK Teams [9] provides team behaviors as BDI entities, with shared knowledge and team plans. Finally, both Madkit [10] and S-MOISE+ [11] are based on concepts of group, role and agent; in which groups represent the context in which agent interactions do take part; whereas roles are seen as positions in the organization that can be fulfilled by agents [2]. However, there still exists a demand for organizational MAS methodologies focused on agent organizations that linked together both analysis, design and implementation phases, employing an organizational agent platform. It would be interesting that services like entering inside an organization or group, registering, querying about its members, and so on, could be implicitly provided by the agent platforms. As the process of creating and modifying an organization is rather complex, we focus our interest in automatizing this process as much as possible, specially regarding all topology aspects and agent communications related with topology.

In the present work we have identified three basic organizational topologies from which other more complex organizations can be composed. This support for MAS organizations has been integrated into SPADE multi-agent system platform [12,13] using Multiuser Conference [14] technology for topology requirements, focusing on agent communication restrictions imposed by each organizational structure. Moreover, we have defined several services related to organization topologies, provided with an API that allows developers to implement MAS organizations and also agents to dynamically create, modify and configure them.

2 Agent Organization Model

Studying both agent organizations [15] and human Organization Theory [16,17] we have identified three basic topologies, that we have named as Organizational Units, from which other more complex organizations can be built. Organizational Units provide both interaction and visibility constraints to agents. These basic topologies are: (i) *Simple Hierarchy*, formed by a supervisor member (that has control over the other members of the unit; centralizes the capture of decisions and coordinates tasks) and several subordinates (that carry out the basic tasks and communicate with each other through the supervisor); (ii) *Team*, in which all members collaborate between them to reach a global and common goal, sharing all their information, and coordination is obtained using mutually accepted decisions and plans; and (iii) *Flat Structure*, that represents an anarchy in which there is not any fixed structure nor control of one member over another, so members are allowed to get information of the existence of other members of their structure. This last unit is mainly needed for modeling more complex structures.

Using the organizational unit concept, more complex and elaborated organizational structures can be designed, such as bureaucracy, matrix, federation, coalition or congregation [15], as it is explained in section 3.

An *agent organization* can be defined as a social entity composed of a specific number of members that accomplish several distinct tasks or functions and that are structured following some specific topology and communication

interrelationship in order to achieve the main aim of the organization. Thus, agent organizations assume the existence of global goals, outside the objectives of any individual agent, and they exist independently of agents [2].

In our model, an *organization* is composed of one or more organizational units and the interactions among agents and units; it has a global goal; needs of social norms for controlling behaviors; and its immerse in a specific environment.

3 Supporting Virtual Organizations with SPADE

We have adapted SPADE multi-agent system platform [12,13] including a *Multi-user Conference* extension [14], which has already been successfully used to communicate agents [13]. SPADE is based in Jabber, an XMPP standard [18] in which servers can be attached to the main Jabber network or be created as independent Instant Messaging sub-networks that do not depend on a centralized network, forming a distributed Jabber network. Following, a brief explanation of how organizational units are modeled in SPADE and how they can be employed to compose more complex organizations is shown. Moreover, the API developed for suporting agent organizational features in SPADE is detailed.

3.1 Organizational Model in SPADE

Organizational Units have been modeled as multi-conference rooms [14] in SPADE, defining its configuration as a tuple $C =< M, A, V, I, S, T >$ where M $\in$ {*Moderated*, *Unmoderated*} represents the moderation feature for controlling which agents can communicate inside the unit; A $\in$ {*Open*, *Members-Only*} shows its accessibility for agents; V $\in$ {*Public*, *Hidden*} represents its visibility; I $\in$ {*Fully-Anonymous*, *Semi-Anonymous*, *Non-Anonymous*} indicates agent identity feature, i.e. whether real agent identity is known by other members of the unit or not; S $\in$ {*Password-Protected*, *Unsecured*} indicates unit access security; and T $\in$ {*Temporary*, *Persistent*} indicates its temporality.

On the other hand, agents have two types of relationships with the organizational units: Affiliation and Unit Role. All agents have only one Affiliation and one Unit Role at each time. *Affiliation* defines long-term relationship of an agent regarding a unit: *owner*, the unit creator with total control over it; *admin*, that controls unit access and belonging; *member*, all agents allowed to be inside the unit; *outcast*, explicitly banned members; and *none*, without any kind of relationship with the unit. *Unit Role* defines the current relationship of the agent within the unit: *moderator*, that controls which agents can communicate inside; *participant*, agents who have been allowed to talk or communicate; *visitant*, agents who are inside but have not been allowed (yet) to communicate with other members of the unit; and *none*, agents who are not inside.

Following, the specific models of basic organizational units are explained.

In a **simple hierarchy** unit [15], a supervisor agent acts as a moderator, allowing and forbidding agents to communicate inside it. This supervisor also controls agent access, assigns tasks and gives orders to subordinates using multichat (messages to everyone) or private messages. Normally, subordinate agents

are not allowed to communicate between them, and do need to communicate through supervisor. Thus, in our model the unit is configured as Moderated and Semi-Anonymous. Moreover, as a hierarchy is a private organization where only authorized agents can enter, the unit is also configured as Members-Only (moderator invitation is mandatory) and Password-Protected (a password offered by the moderator is required). In this way, the possibility of an unauthorized agent to participate and interfere in the processes of the hierarchy is blocked. Therefore, the simple hierarchy unit configuration is set as C=(*Moderated*, *Members-Only*, V, *Semi-Anonymous*, *Password-Protected*, T) where V ∈ {*Public*, *Hidden*} and T ∈ {*Temporary*,*Persistent*}.

A **team** is normally composed of a closed set of agents which have a common goal, share all needed information and whose tasks are assigned to the most qualified members of the group [15]. There is not any hierarchical supervisor, although there could be a representant or leader of the group, in charge of recruiting new agents and resolving some task assigning conflicts. Agents are in constant communication with each other, both with multichat way or with private messages. Thus, in our model, the team unit configured as C=(*Unmoderated*, *Members-Only*, V, *Non-Anonymous*, S, T) where V ∈ {*Public*, *Hidden*}; S ∈ {*Password-Protected*, *Unsecured*} and T ∈ {*Temporary*, *Persistent*.

In the **flat unit**, there is a complete anarchy, as no agent can have any control over the others. Therefore, anyone can enter inside this unit and communicate with others. In our model, its configuration is set as C=(*Unmoderated*, *Open*, *Public*, *Non-Anonymous*, *Unsecured*, *Persistent*).

3.2 Modeling Complex Agent Organizations

More complex and elaborated organizational structures can be built using organizational units, as detailed as follows.

A **bureaucracy** is an organization composed of layered sets of simple hierarchies in which subordinates of a higher level are also supervisors of an inferior level. The top level is in charge of controlling that the main goals of the organization can be accomplished. In the bottom levels, worker agents carry out basic tasks. In all intermediate levels, agents filter information to higher levels, receive orders and take decisions based on those orders and their own knowledge; and have control over lower levels. Therefore, authority is centralized and decision-making follows a chain of command.

In our model, a bureaucracy is designed by relating simple hierarchy units through each other, like a tree-based structure. Initially, when defining the organization, a top level simple hierarchy is created and then more hierarchy units can be added, in which subordinate members of higher levels act as supervisors or moderators of lower levels.

A **matrix** is an organization that combines a functional structure (in which tasks and units are grouped based on technical specialization or business functions) and a divisional structure (in which units are grouped by manufacturing products, types of clients, geographical areas, etc., being autonomous groups that contain all functions needed) [16]. Therefore, members have two supervisors:

the functional manager (with control over functional tasks) and the product manager (with control over final products).

In our model, a matrix is represented as an organization composed by an initial team unit, in which all functional and divisional supervisors are included; and several hierarchical units with one functional or divisional supervisor acting as moderator, and worker agents as subordinates.

A **federation** represents a group of agents that have given some amount of autonomy to a single delegate which represents the group [15]. Member agents can only interact with this representative agent, who might act as task allocator, monitor progress, broker or facilitator and it also interacts with agents outside the unit. Therefore, federations are normally connected to each other through their representative agents.

In our approach, a federation is modeled as an organization composed of several hierarchies, whose supervisors are also members of a team unit, configured as C=(*Unmoderated*, *Members-Only*, *Hidden*, *Non-Anonymous*, *Password-Protected*, *Persistent*). When creating a federation, the team unit is established and then hierarchical units can be added dynamically as needed.

A **coalition** represents a temporal group of agents that work together to reach a common goal [15]. Agents join the coalition to obtain several benefits and reduce costs when working together. The coalition is dissolved when its goal is achieved; when it is not needed anymore; or a critical mass of agents departs.

In our approach, the organizational structure of a coalition is modeled as a team, in which there is an admin agent in charge of controlling the minimum number of agents needed for the coalition and allowing members to participate and communicate inside it when the coalition is formed. This admin agent can also act as a representative and intermediary of the group as a whole. Therefore, coalitions are composed of one or more dynamic groups, created when new task needs appear. Those units must be Temporary, in order to be destroyed once the coalition is dissolved. Moreover, they must be Open, Public, Unsecured and Non-Anonymous so anyone can enter inside the coalition and communicate with all members of the unit. Finally, they must be Moderated, allowing the admin agent (acting as moderator) to enable or forbid agent conversations depending on the critical number of members. It can also send agents off once the coalition is dissolved. Therefore, a coalition is represented as an organization composed of dynamic team units configured as C=(*Moderated*, *Open*, *Public*, *Non-Anonymous*, *Unsecured*, *Temporary*).

A **congregation** represents a group of agents willing to collaborate with each other in long-term relationships . Agents have similar or complementary features and have banded together to facilitate the process of finding suitable collaborators [15]. Internally, a congregation has not any predefined structure, so it can be designed with flat, team or simple hierarchy organizational units. Members can join and leave those units dynamically, but they only belong to one of them at a specific time. In our model, a congregation is represented as an organization composed of flat organizational units, based on a specific theme or goal, in which anyone can enter and have both private or multichat conversations with

other members. Units can also remain in the system even when all its members are gone. Moreover, a minimum number of members is normally established in order to obtain benefits of a congregation, so a moderator agent might control when communications are allowed. Thus organizational units for a congregation are configured as C=(M, *Open*, *Public*,*Non-Anonymous*, *Unsecured*, *Persistent*), where M ∈ {*Moderated*, *Unmoderated*}.

3.3 Organizational Unit Services in SPADE

As stated before, current agent platforms rarely support agent organizational features, so designers need to implement on their own many services related with organization dynamics, such as agent registering, entrance or exit and so on. Thus, we have included in SPADE a series of services related to an organizational unit, covering aspects of agent interaction control, unit member control and organizational unit control. Table 1 shows a brief description of the API developed for those services.

Agent interaction control. In SPADE, agents have a real, unique and global identifier (AID) valid for the entire system. They also have a nickname in every organizational unit in which they are immersed. Such nickname must be unique within the unit, and can be changed dynamically every time the agent enters,

Table 1. Brief description of API functions for SPADE MAS platform. Terms used are: o=Organization object; u=Organizational Unit object.

API FUNCTION	DESCRIPTION
u.AddAdmin(AgentNameList)	Adds specified agents to the admin list
u.AddModerator(AgentNameList)	Adds specified agents to the moderator list
u.AddOwner(AgentNameList)	Owner agent adds specified agents to the owner list
u1=o.AddUnit(Name,Type,GoalList, AgentList)	Creates a new unit inside an organization
u1=u.AddUnit(Name,Type,GoalList, AgentList)	Creates a new unit hierarchically dependent on another
u.BanAgent(AgentNameList)	Specified agents are not allowed to enter again
u.Destroy()	Organization owner destroys a unit
GoalList=u.GetGoal()	Returns a list of goals
u.GetMaxAgents()	Maximum agents allowed to enter inside the unit
AgentNameList = u.GetMemberList()	Returns a list with agents' names belonging to the unit
u.GetMinAgents()	Minimum agents needed to allow conversations inside
u.GetNumberOfAgents()	Returns current number of agents that are inside
OrgList=self.GetOrganizationList()	Returns names of current organizations
DF=o.GetRegistrationForm(UnitName)	Returns a dataform with joining requested information
UnitDescription=o.GetUnitInfo(Name)	Returns a unit description, including its type and goals
UnitList=o.GetUnits()	Returns names of public units of a specific organization
u.GiveVoice(AgentNameList)	Adds names of specified agents to participant list
u.Invite(AID,password)	Sends an invitation to an agent for joining the unit
o=self.JoinOrganization(OrgName)	Agent joins a specific organization
u=o.JoinUnit(UnitName, AgentName)	Agent requests joining a specific unit
u.KickAgent(AgentNameList)	Specified agents are expulsed from the unit
u.LeaveUnit()	Agent leaves and it is removed from the member list
o=Organization(Name, Type, Goals, Agents, CL)	Creates an organization and its initial unit
u.RemoveAdmin(AgentNameList)	Removes specified agents from the admin list
u.RemoveModerator(AgentNameList)	Removes specified agents from the moderator list
u.RemoveOwner(AgentNameList)	Removes specified agents from the owner list
u.ReserveName(AgentName)	An agent reserves its name so nobody else can use it
u.RevokeVoice(AgentNameList)	Removes agents from participant list
u.SendMessage(Message)	Sends a message to unit members through its server
u.SendPrivateMessage(RecName,Message)	Sends a message to receiver agent through unit server
o.SendRegistrationForm(UnitName, DataForm)	Sends dataform for a specific unit
u.SetGoal(GoalList)	Updates unit goals
u.SetMaxAgents(number)	Maximum agents allowed to enter inside the unit
u.SetMinAgents(number)	Minimum agents needed to allow conversations inside

or even when it is already inside. Nickname reservation is also allowed for an agent to hold the same nickname every time it enters. AIDs are used for the sending of private messages in a one-to-one fashion, even between agents in different units. Nicknames also allow to send private messages, but both sender and receiver must be on the same unit, whose server is in charge of establishing the appropriate connections between agents. Finally, organizational units allow multichat typed messages, in which an agent sends a message directly to the unit server, who relies it to all its members.

When entering, agents communicate their nickname to the unit server through a presence-type message. Then, the server relies it to the rest of occupants. In a Non-Anonymous type, server also adds agent AID to such message. In a Semi-Anonymous type, it only gives this AID to moderator agents. Moreover, if *presencebroadcast* parameter of the unit configuration form is set to *moderator* value, then presence messages are only relied to moderator agents. This configuration is the default on simple hierarchies. In this way, moderators (supervisors) would be the only ones in the know of the AID of the other agents, so subordinates are not capable to send private messages to each other.

Services offered for agent interaction control are (table 1):

- Enable / disable communications: in Moderated units, only *participants* have the right to send messages, as private or multichat messages. Moderator agents can control communications by means of modifying who acts as participant using *u.GiveVoice(AgentNameList)* and *u.RevokeVoice(AgentNameList)*.
- Bilateral interactions: an agent can send direct messages to another using its AID or through the unit server, using its nickname and *u.SendPrivate Message(ReceiverName, Message)* function. Those interactions can also be restricted, preventing agents to know the AID of others (Fully-Anonymous or Semi-Anonymous units), or even the nicknames of other agents (limiting the presence broadcast to moderator agents).
- Multiple interactions (to everyone): an agent can send a message to the unit server, using *u.SendMessage(Message)* function, and it would be relied to all its members. The agent does not need to know who its occupants are.

Unit Member Control. An agent can enter an Open unit freely, but it needs to be invited or registered for a Members-Only one. In this case, the monitor agent can send it a specific invitation (supplying the password, if it is a Password-Protected type), by means of *u.Invite(AID, password)* function. In case of registration, an agent first sends a submission request to the unit and then receives a data form to fill in, using *o.GetRegistrationForm(UnitName)* function. This data form can be adapted in the implementation of the system and will normally require nickname, agent role, known ontologies and communication languages. Once filled, the agent sends it to the requested unit through the organization, using *o.SendRegistrationForm(UnitName, DataForm)* function. This data form can be lately reviewed by an admin agent, who decides whether accept the registration process or not. If so, the agent is included in the member list and allowed to enter, which will occur when the agent invokes *o.JoinUnit(UnitName, agentname)*.

Services offered for controlling Organizational Units are (table 1):

- Agent Identity: agents can request who are the other members of the unit using *u.GetMemberList*, which provides agent AIDS or nicknames, depending on configuration.
- Admission: in an Open unit, agents can enter directly. In Members-Only ones, agents must be previously included in the member list by the admin agent, who can send them invitations to join using *u.Invite(AID,password)*.
- Registration: agents fill in a data form that should be approved by the admin agent, using both *o.GetUnitRegistrationForm(Name)* and *o.SendRegistrationForm(Name,DF)* functions. Moreover, once inside, an agent can also reserve its nickname with *u.ReserveName(AgentName)*, so it is always used by the same agent.
- Exit: any agent can leave the unit whenever it desires using *u.LeaveUnit()* function, and it is removed from the member list.
- Expulsion: moderator agents can send off any member of the unit when necessary (for example, when a member has violate a social norm), using *u.KickAgent(AgentName)* function or *u.Ban(AgentName)*. In last case, agents will not be allowed to enter into the unit again.
- Quantity of members: *u.GetMinAgents()*, *u.GetMaxAgents()* and *u.GetNumberOfAgents()* provide actual restrictions about maximum and minimum quantities of members and its current number.
- Cloning representatives: monitor, admin and owner agents can delegate their functions to other members in a dynamic way, allowing for a greater flexibility and robustness of the system, by means of several functions such as *u.addModerator(AgentNameList)*, *u.addOwner(AgentNameList)* or *u.addAdmin(AgentNameList)*.

Organizational Unit Control. In SPADE, an agent can be initially run in the MAS platform without belonging to any organization. Then, it can create new organizations or request which are the organizations of the system and join one of them. By default, any agent can create a new organization using its constructor *o=Organization(Name,Type,GoalList,AgentList,ContentLanguage)*. This service can be restricted to some specific agents during the implementation of the system, though. Creating an organization internally produces that the organization name is registered in the DF server and a new organization unit is created, with initial configuration defined by the organization type. Moreover, the agent is set as owner of both organization and unit. And invitation messages are sent to all required members (specified in the AgentList).

Once an organization is defined, its owner can add more organizational units inside with both *o.addUnit(Name,Type,GoalList,AgentList)* or *u.addUnit(Name, Type, GoalList,AgentList)* functions. In the last case, branched-related units are created. Only coherent types with the organization structure are allowed.

Owner agents can also dynamically modify the unit configuration, requesting a configuration form to the server and filling in it with all new requirements. Those changes will be done if they are permitted by the organization type.

They can even affect to some members of the unit. For example, if it becomes a Members-Only one, then all non-members must be kicked out.

Services offered for Organizational Unit control are (table 1):

- Information about existing organizations: any agent would be able to query which are the organizations of a system using *GetOrganizationList()*.
- Join an organization: agents can enter inside an organization by means of *JoinOrganization(Name)* function. Then they will belong to the initial organizational unit, but with the lowest permissions. They will not be allowed to participate inside the organization until they join a specific unit.
- Information about existing units: agents inside an organization would be able to query which are the public organizational units and their configurations, using both *o.GetUnits()* and *o.GetUnitInfo(UnitName)* functions.
- Creation of new units: dynamic creation of new ones is restricted to the organization owner. Besides, facilities to create given type of units are provided, using *o.addUnit()* function.
- Automatic deletion of units: Temporary type units are automatically eliminated when they become empty. Persistent ones remain indefinitely until their owner removes them.
- Deletion of units: the organization owner can destroy units if needed, using *u.Destroy()*. All their members will be kicked off.
- Configuration changes: owners can make dynamic modifications of unit configurations, requesting a form to its server and filling in it with all new requirements.
- Unit goal: goals of the unit can be established or modified using *u.GetUnitGoal* and *u.SetUnitGoal*. It is also possible to restrict who is entitled to modify it.

4 Conclusions

In this work we have applied Organizational Theory concepts in SPADE multiagent system platform to give support to agent organizations. Starting with the three identified basic organizational topologies, we have established the ways of performing and configuring several kinds of organizations using advanced features of *Multi-User Conference*. We have also defined several services offered by SPADE to ease the implementation of organization dynamics.

We are now working on a MAS prototype for intelligent management of tourist sites using the SPADE platform, in which different organizational structures can be compared. All this work is integrated into a higher project whose driving goal is to develop a new way of working with agent organizations, including both a MAS platform (SPADE) which provides a wide range of services related to agent organizations, and a new methodology for the development of MAS based on concepts found on human organizations.

References

1. Dignum, V., Meyer, J., Wiegand, H., Dignum, F.: An organization-oriented model for agent societies. In: Lindemann, G., Moldt, D., Paolucci, M. (eds.) RASTA 2002. LNCS (LNAI), vol. 2934, Springer, Heidelberg (2004)
2. Dignum, V., Dignum, F.: A Landscape of Agent Systems for the Real World. Tech. Report Utrecht University (2007)
3. Argente, E., Giret, A., Valero, S., Julian, V., Botti, V.: Survey of MAS methods and platforms focusing on organizational concepts. Frontiers in Artificial Intelligence and Applications 113, 309–316 (2004)
4. Argente, E., Julian, V., Botti, V.: Multi-agent system development based on organizations. ENTCS 150, 55–71 (2006)
5. Dignum, V.: A Model for Organization Interaction: based on Agents, founded in Logics. SIKS Dissertation Series 2004-1. Utrecht University, PhD Thesis (2004)
6. Dignum, V., Vazquez, J., Dignum, F.: OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Programming Multi-Agent Systems. LNCS (LNAI), vol. 3346, Springer, Heidelberg (2005)
7. Hubner, J.F., Sichman, J.S., Boissier, O.: MOISE+: Towards a structural, functional and deontic model for MAS Organization. In: Alonso, E., Kudenko, D., Kazakov, D. (eds.) Adaptive Agents and Multi-Agent Systems. LNCS (LNAI), vol. 2636, Springer, Heidelberg (2003)
8. Esteva, M., Rosell, B., Rodriguez, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) Adaptive Agents and Multi-Agent Systems II. LNCS (LNAI), vol. 3394, pp. 236–243. Springer, Heidelberg (2005)
9. Howden, N., Ronnquist, R., Hodgson, A., Lucas, A.: JACK Intelligent Agents-Summary of an Agent Infrastructure. In: 5th Int. Conf. on Autonomous Agents (2001)
10. Ferber, J., Gutkenecht, O.: A meta-modelfor the analysis and design of organizations in multi-agent systems. In: Proc. ICMAS98, IEEE CS Press, Los Alamitos (1998)
11. Hubner, J.F., Sichman, J.S., Boissier, O.: A Middleware for developing Organised Multi-Agent Systems. In: Sichman, J.S., Antunes, L. (eds.) MABS 2005. LNCS (LNAI), vol. 3891, pp. 64–78. Springer, Heidelberg (2006)
12. Escrivà, M., Palanca, J., Aranda, G., García-Fornes, A., et al.: A Jabber-based multi-agent system platform. In: Proc. of AAMAS06, pp. 1282–1284 (2006)
13. Palanca, J., Escrivà, M., Aranda, G., García-Fornes, A., Julian, V., Botti, V.: Adding new communication services to the fipa message transport system. In: 4th German Conference on Multiagent System Technologies, pp. 1–11 (2006)
14. Saint-Andre, P.: XEP-0045: Multi-User Chat. Technical report, Jabber Software Foundation (September 13, 2006), `http://www.xmpp.org/extensions/xep-0045.html`
15. Horling, B., Lesser, V.: A survey of multiagent organizational paradigms. The Knowledge Engineering Review 19, 281–316 (2004)
16. Robbins, S.P.: Organizational Behavior. Prentice-Hall, Englewood Cliffs (2003)
17. Fox, M.: An organizational view of distributed systems. IEEE Trans. on System, Man and Cybernetics 11, 70–80 (1981)
18. Jabber Software Foundation: Extensible Messaging and Presence Protocol (XMPP): Core. Tech. Report (2004), `http://www.ietf.org/rfc/rfc3920.txt`

The Agents' Attitudes in Fuzzy Constraint Based Automated Purchase Negotiations*

Miguel A. Lopez-Carmona, Juan R. Velasco, and Ivan Marsa-Maestre

Department of Automatica, University of Alcala
Edificio Politecnico, Ctra. NII, km. 33.600
28871 Alcala de Henares (Madrid), Spain
Tel.: +34 918856673, Fax:+34 918856641
{miguelangel.lopez,juanramon.velasco,ivan.marsa}@uah.es

Abstract. In this paper we present an experimental analysis of a fuzzy constraint based framework for automated purchase negotiations in competitive trading environments. The main goal of this work is to show by experimental analysis which combination of different agents' negotiation attitudes allows buyer and seller agents to improve the multi-attribute bilateral negotiation processes. Agents' attitudes are related to the agents' strategic behaviour in the negotiation process, where strategic behaviours are described in terms of expressiveness and receptiveness.

1 Introduction

Automated negotiation is an important challenge in the Multi-Agent Systems community which has been covered from different areas such as game theory [1] and distributed artificial intelligence [2,3]. This paper focuses on multi-attribute bilateral negotiation using fuzzy constraints in purchase negotiation scenarios. In our previous work [4,5], we present a fuzzy contraint based model for automated purchase negotiations, and show how with our approach the negotiation processes are more efficient than with previous approaches [6] where mainly positional bargaining is used. The negotiation model uses fuzzy constraints [7] to capture requirements and to express proposals, and the proposed interaction protocol is a dialogue game protocol [8,9] where a set of locutions and decision mechanisms which fire them are fully specified. The aim of this work is to show how the different attitudes in terms of expressivity and receptivity of the negotiating agents influence the reached agreements. For the seller agent, an expressiveness parameter controls whether the seller agent expresses its preferences for a specific relaxation of the previous buyer's demands, while a receptivity parameter modulates the seller's attitude regarding the buyer's purchase requirements. For the buyer agent, an expressiveness parameter controls the use of purchase requirement valuations, while a receptivity parameter modulates the buyer's attitude regarding a relax requirement received from a seller agent.

* This work has been supported by the Spanish Ministry of Education and Science grant TSI2005-07384-C03-03.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 246–255, 2007.

The rest of the paper is organized as follows. Section 2 describes the basic notation used through the paper and the negotiation framework mentioned before. Section 3 presents a scenario for the experiments and the results obtained. And finally, Section 4 shows the conclusions.

2 Negotiation Framework

Our negotiation framework is based on a fuzzy constraint satisfaction problem framework, and it is comprised of a buyer and seller agents' domain knowledge, and a complex interaction protocol based on dialogue games [8]. A dialogue game is described as a set of locutions, a set of decision mechanisms, and a set of transition rules which states how different locutions invoke decision mechanisms, and how different decision mechanisms utter locutions. For reasons of space only those mechanisms involved in bargaining tasks within the negotiation processes are described. First, the fuzzy constraint satisfaction problems framework and the basic domain knowledge of the agents are presented.

A *fuzzy constraint satisfaction problem* (*FCSP*) is a 3-tuple $\left(X, D, C^{\mathrm{f}}\right)$, where $X = \{x_i| = 1, ..., n\}$ is a finite set of variables, $D = \{d_i| = 1, ..., n\}$ is the set of domains of the variables, and $C^{\mathrm{f}} = \{R_i^{\mathrm{f}} | i = 1, ..., m\}$ is a set of fuzzy constraints over the variables. The function that indicates how well a given constraint is satisfied is the *satisfaction degree function* $\mu_{R_i^{\mathrm{f}}}$, where 1 indicates completely satisfied and 0 indicates not satisfied at all. Using a cut-set technique and given the *cut level* $\sigma \in [0, 1]$, the induced *crisp constraint* of a fuzzy constraint R^{f} is defined as R^{c}. Finally, the *overall (or global) satisfaction degree* of a potential solution v_X is $\alpha(v_X) = \oplus \left\{\mu_{R^{\mathrm{f}}}(v_X) | R^{\mathrm{f}} \in C^{\mathrm{f}}\right\}$, where $\oplus$ is an aggregation from $[0, 1]^m$ to $[0, 1]$. Buyer agent's requirements on the attributes of a product are described by means of a FCSP. On the other hand, a negotiation profile $N_{\mathrm{b}} = \{\xi, \eta\}$ describes the buyer agent's attitude, where $\xi \in \{0, 1\}$ controls the use of purchase requirement valuations, and $\eta \in [0, 1]$ modulates the buyer's attitude regarding a relax requirement received from a seller agent. A *purchase requirement* is defined as $\lambda_{\mathcal{B}_{\mathrm{req}}} = \bigcap \left\{R_i^{\mathrm{c}(\sigma_i)}\right\}$, where $R_i^{\mathrm{c}(\sigma_i)}$ is a *crisp constraint* induced from R_i^{f} at a cut level σ_i. Given a purchase requirement, the *potential overall satisfaction degree* (*posd*) the buyer may get if the requirement is met is $\alpha^{\lambda_{\mathcal{B}_{\mathrm{req}}}} = \oplus \{\sigma_i | i = 1, ..., m)\}$, where σ_i represents the cut level applied to constraint R_i^{f}. Finally, given a $\lambda_{\mathcal{B}_{\mathrm{req}}}$, a *purchase requirement valuation* is defined as a set $v_{\mathcal{B}_{\mathrm{req}}} = \{v_i | v_i \in [0, 1]\}$, where v_i is the degree of importance that the constraint i has for the buyer agent. On the other hand, a seller agent owns a catalogue of products $S = \{s_j | s_j = (p_j, u_j)\}$, where p_j is the vector of attributes and u_j is the profit the seller agent obtains if the product is sold. A negotiation profile $N_{\mathrm{s}} = \{\psi, \beta\}$ describes the seller agent's attitude, where $\psi \in \{0, 1\}$ controls whether the seller agent uses relaxation requirements in order to express its preferences for a specific relaxation of the previous buyer's demands, and $\beta \in [0, 1]$ modulates the seller's attitude regarding a purchase requirement received from a

buyer agent. A *relaxation requirement* is defined as a set $\rho_{\mathcal{B}_{\text{req}}} = \{r_i | r_i \in [0,1]\}$, where r_i is the preference for constraint i to be relaxed.

Now, the following mechanisms are described according to the type of participants: Buyers (**B**) or Sellers (**S**).

B1: Generate Purchase Requirement enables the buyer agent to generate a new purchase requirement.

Given a purchase requirement $\lambda^k_{\mathcal{B}_{\text{req}}}$ previously submitted to a seller agent at instant k, the mechanism obtains the set $\{\alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}}_{R^{\text{f}}_i}\}$, where $\alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}}_{R^{\text{f}}_i}$ represents the *posd* if constraint R^{f}_i from $\lambda^k_{\mathcal{B}_{\text{req}}}$ is relaxed, and computes its maximum. The set of potential purchase requirements with a *posd* equal to the maximum is described as $\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}$, so this set represents the new purchase requirements which minimize the lost of *posd*. Finally, the mechanism applies the *constraint selection function* csf which is defined as: $csf = \arg(\max_{\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}} \alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}}_{R^{\text{f}}_i} + r_i * \eta)$ in order to select the purchase requirement from $\{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}\}_{\text{feasible}}$. We can see how the η parameter in the csf function modulates in which degree the buyer agent attends the seller agent's requirements in order to select the constraint that will be relaxed to generate the new purchase requirement $\lambda^{k+1}_{\mathcal{B}_{\text{req}}}$.

B2: Generate Purchase Requirement Valuation enables the buyer agent to generate the valuation of a purchase requirement which is going to be submitted to a seller agent.

If $\xi = 0$ this mechanism does not work. If $\xi = 1$ this mechanism makes a valuation of the purchase requirement generated in the previous mechanism. In our experiments we have defined a linear function where those constraints that when relaxed generate a low *posd* are more valued than constraints which generate a higher *posd*.

S1: Generate Potential Sale Offers makes a selection of products which the seller agent considers as good candidates for a sale offer.

If a seller agent cannot satisfy a purchase requirement, it may encourage the buyer agent to change its proposals. A good alternative to promote a convergence in the negotiation is to express how the purchase requirement should be relaxed. We plan to do this by means of a *relaxation requirement*. This mechanism considers the first task in this process: the selection of candidates for a future sale offer. We have identified two main aspects when making this selection: the *local utility*, which depends on the u_j parameter (i.e. the profit), and the *viability*, which depends on the similarity between the p_j candidate and the purchase requirement $\lambda^k_{\mathcal{B}_{\text{req}}}$, as well as on the expected buyer's valuation. A seller agent may give more or less importance to each aspect by means of the $\beta \in [0,1]$ parameter. The function $prefer$ estimates the goodness of a potential sale offer in terms of *utility* and *viability*:

$$prefer(p_j) = \beta * u_j + (1-\beta) * \hat{viability}(p_j, \lambda^k_{\mathcal{B}_{\text{req}}}, \upsilon_{\mathcal{B}_{\text{req}}}) \ .$$

In our experiments the $prefer$ function is defined as follows:

$$prefer(s_j) = \beta * u_j + (1 - \beta) * (1 - sqrt(\sum_{i=1}^{n} (\hat{dist}(a_{ji}, \lambda^{t,i}_{\mathcal{B}_{req}}) * v_{\lambda^{t,i}_{\mathcal{B}_{req}}})^2/n))$$

where $\hat{dist}(a_{ji}, \lambda^{t,i}_{\mathcal{B}_{req}})$ represents a per-attribute estimate of distance, and $v_{\lambda^{t,i}_{\mathcal{B}_{req}}}$ is the per-attribute seller agent's valuation. Once the $prefer$ function has been computed for all the products in the catalogue, those products with a value exceeding a threshold are selected. This is the set S_p.

S2: Generate Relax Requirement generates the relax requirement $\rho^k_{\mathcal{B}_{\text{req}}}$ to submit to the buyer agent, where $r_i = 1$ if constraint R^{f}_i in $\lambda^k_{\mathcal{B}_{\text{req}}}$ has not been satisfied for any product in S_p, and otherwise $r_i = 0$.

3 Experiments

Our negotiation framework allows us to test with different expressive and receptive strategies. An expressive buyer agent ($\xi = 1$) makes use of purchase valuations, whilst a non-expressive one ($\xi = 0$) does not. The receptiveness is determined by η, which has a continuous domain that we limit to non-receptive ($\eta = 0$) and receptive ($\eta = 1$). Finally, we have defined the calculation of the *posd* operator as $\otimes = min$. On the other hand, an expressive seller agent ($\psi = 1$) makes use of relax requirements, whilst a non-expressive one ($\psi = 0$) does not. The receptive profile determined by β has a continuous domain that we limit to 0, 0.5 and 1 (value 0 indicates no receptivity, 0.5 intermediate, and 1 maximum). However, not all the combinations of strategies are valid.

3.1 Analysis of Validity of Individual Strategies

We look first at agent level and we begin with the **buyer agent**.[1]

(BAer) expressive and receptive, **(BAner) non-expressive and receptive**, and **(BAnenr) non-expressive and non-receptive** are valid strategies. However, **(BAenr) expressive and non-receptive** strategies makes no sense as the purpose of the valuation is to redirect the negotiation in such a way that the seller agent sends useful relax requirements. If the agent does not analyse relax requirements, the valuation is of no use whatsoever. To sum up, the buyer agent can behave in three different ways: **BAer**, **BAner** y **BAnenr**.

We now analyse the **seller agent**.

[1] To denominate the different strategies we use the following convention. **BA** refers to a Buyer agent Attitude, and **SA** refers to a Seller agent Attitude. Next, first the types of expressive behaviour **ne** and **e** appear, to define non-expressivity and expressivity respectively. Finally the types of receptive behaviour **nr** and **r** appear, to define non-receptive and receptive respectively. Only for a receptive behaviour (**r**) is a numerical suffix added to consider the level of receptivity.

(SAer1) expressive and receptive ($\beta = 0$), **(SAer0.5) expressive and receptive** ($\beta = 0.5$), **(SAenr) expressive and non-receptive** ($\beta = 1$), and **(SAnenr) non-expressive and non-receptive** are valid strategies. However, **(SAner1 or SAner0.5) non-expressive and any receptive strategy** makes no sense as a non-expressive seller agent does not send relax requirements, and the main purpose of a receptive strategy is to direct the relax requirements. To sum up, the seller agent can behave in accordance with four different strategies: **SAer1**, **SAer0.5**, **SAenr** and **SAnenr**.

3.2 Analysis of Validity of Combination of Strategies

There are a total of 12 possible combination of strategies. However, some of these combinations are not coherent. In **BAer vs SAnenr** the buyer agent's valuations are not taken into account by the seller agent, which furthermore is not expressive. This aspect is detectable by the buyer agent, given that it does not receive relaxation requirements. A rational agent will not send valuations if it knows they are of no use. This combination is not stable and therefore equilibrium is not possible. The best strategy for a buyer agent under these circumstances is to change to a non-expressive and non-receptive strategy **BAnenr**. In **BAner vs SAnenr** neither of the agents is expressive, so for the buyer agent to be receptive makes no sense, and furthermore this fact it detectable by the buyer agent. A rational buyer agent would change to a **BAnenr** strategy. After this analysis, there are 10 pairs of balanced strategies : **BAer vs SAer1**, **BAer vs SAer0.5**, **BAer vs SAenr**, **BAner vs SAer1**, **BAner vs SAer0.5**, **BAner vs SAenr**, **BAnenr vs SAer1**, **BAnenr vs SAer0.5**, **BAnenr vs SAenr**, y **BAnenr vs SAnenr**. To simplify this repertoire we have made the following groupings:

BAer vs SAexrx. This group has in common the fact that the buyer agent is simultaneously expressive and receptive, and the seller agent is expressive. Furthermore it seems obvious that the seller agents' different receptive profiles will affect the results of the negotiation, because the generation of relax requirements varies depending on this profile. Therefore, a priori we need to test with the three combinations that make up the group.

BAner vs SAexrx. This group of strategies has in common the fact that the buyer agent is not expressive, but it is receptive and the seller agent is expressive. When the seller agent is receptive, intuitively we can affirm that the results of the negotiations are different to those of the previous group. This is so because the buyer agent's valuations are not available to the seller agent. However, when the seller agent is not receptive the scenario is identical to the previous group. In other words, if the seller agent is not receptive it makes no difference whether the buyer agent sends valuations or not. In conclusion, the **BAner vs SAenr** pair is identical to **BAer vs SAenr**, as far as the results of the negotiation are concerned. To speed up the tests of this type, we have opted to define the test as **BAner vs SAenr**.

BAnenr vs SAxexrx. This group of strategies is characterised by the non-expressivity and non-receptivity of the buyer agent. So, it makes no difference whether the seller agent is expressive or receptive or not as the buyer agent will be unable to take it into account. To speed up the execution of the tests in this group, we have opted to define as representative the **BAnenr vs SAnenr** pair.

Summarizing, there exist 6 pairs of strategies that can resolve negotiations with disparate results.

3.3 Buyer's Preferences and Seller's Catalogue of Products

The buyer agent's preferences are described as a 5 fuzzy constraint problem $R^f_{1...5}$ over 5 attributes $a_{1...5}$. Given a catalogue of products S the set of products that may be a solution to a negotiation is the *solution set* $S_{sol} \subseteq S$. This set is comprised of the products which maximize the buyer agent's utility. This occurs when the seller agent is non-strategic with regards to the occultation of products and the buyer agent relaxes constraints minimizing the lost of *posd*. Finally, the *noise set* $S_{noise} \subseteq S$, is the set of complementary products to S_{sol}, so that $S_{sol} \cup S_{noise} = S$. In the experiments the set S_{sol} is defined as a set of products where $\alpha(p_i) = 0.7$ for the buyer agent, while the utilities for the set S_{noise} are 0.1, 0.2 or 0.3. Once the products from the solution and noise sets have been generated, the next step is to assign utility values u_j to each of them. In the case of S_{noise} these are generated randomly using a uniform allocation between 0.9 and 1, while for $S_{solution}$ a uniform allocation between 0 and 0.69 is used. To test the single point efficiency of the negotiations, also randomly, the utility of one of the products from the set S_{sol} is assigned 0.7. The aim is to see if this solution is reached after a negotiation. With these utilities, the seller agent's preferred sale offers are the noise set products. However, an intelligent seller agent, would come to the conclusion that these products are not a valid sale offer and it would concentrate on obtaining the best solution from amongst those products that can really be a solution, in other words, from the solution set.

3.4 Test Results

For each of the 6 pairs of strategies that we analyse and the different sizes of catalogues, 300 negotiations are carried out. We take as a reference the number of products from the solution set, so that the noise set is the same size as the solution set in every case. Taking into account that the buyer agent's overall satisfaction degree is known, the result we need to analyse is the utility the seller agent obtains from each negotiation. The results show the median and the success rate, where the success rate estimates the number of times that the pareto-optimal solution is obtained, that is to say, the solution in which $u_j = 0.7$. In every case the calculated confidence interval is 95%. The summary results are shown in Table 1.

It can be seen that with the **BAnenr vs SAnenr** strategies the success rate stabilizes around 10%, although for 4 and 8 products the rate is higher, which is logical, as the number of relax combinations is greater than the number of products. There is a dip in the median value with 16 products. However

Table 1. Summary of Results

BAnenr vs SAnenr, BAer/BAner vs SAer1, BAner vs SAenr, BAer vs SAer0.5			
Number of products	Success rate	Confidence interval	Median
4	0.4400	0.3830 0.4982	0.6432
8	0.32	0.2676 0.3760	0.5502
16	0.15	0.1116 0.1955	0.4867
32	0.12	0.0855 0.1622	0.5362
64	0.15	0.1116 0.1955	0.6326
128	0.11	0.0769 0.1510	0.6419
256	0.07	0.0439 0.1050	0.6686
BAner vs SAer05			
Number of products	Success rate	Confidence interval	Median
4	0.9500	0.9189 0.9717	0.7
8	0.7600	0.7076 0.8072	0.7
16	0.6900	0.6343 0.7419	0.7
32	0.5200	0.4618 0.5778	0.7
64	0.4500	0.3928 0.5082	0.6859
128	0.3600	0.3056 0.4172	0.6647
256	0.3300	0.2770 0.3864	0.6807

the median grows again as the number of products increases. This effect was foreseeable, as when the number of products is augmented, the probability that the seller agent has products with a high utility for the purchase requirement is greater. The results for the **BAer/BAner vs SAer1** strategies are similar to the **BAnenr vs SAnenr** strategies. This result was foreseeable, taking into account that the seller agent is not an utility maximiser. In the **BAer vs SAer0.5** strategies the valuation of the purchase requirements distracts the seller agent, so it unfavourably modifies the preferences of the different sale offers and ends up concentrating the search in the noise set, and the results are similar to the **BAnenr vs SAnenr** strategies. The **BAner vs SAenr** strategies are also similar to those obtained with the **BAnenr vs SAnenr** strategies. These results are very important, because they allow us to appreciate how, when a seller agent limits itself to looking out for its best interest and only thinks about the utility, the results are not good. Finally, with the **BAner vs SAer0.5** strategies we can check how the results are significantly better in every case. This test shows that the expressivity of the seller agent is key to obtaining satisfactory solutions.

In the Figure 1 the results obtained from the tests of the BAnenr vs SAnenr, and BAner vs SAer05 strategies are summarised. In the top graphic the medians are shown, and in the bottom one the success rates. The success rates follow the same trend for all the catalogues, with a noticeable improvement in the case of the BAner vs SAer05 strategies. As regards the medians, for catalogues with up to 64 products, the results are optimum. For catalogues with more than 256 products the strategies tend to converge, so expressivity is not a determinant factor. It should be recalled that a heavily populated catalogue means the seller agent will have high utility sale offers with a higher probability.

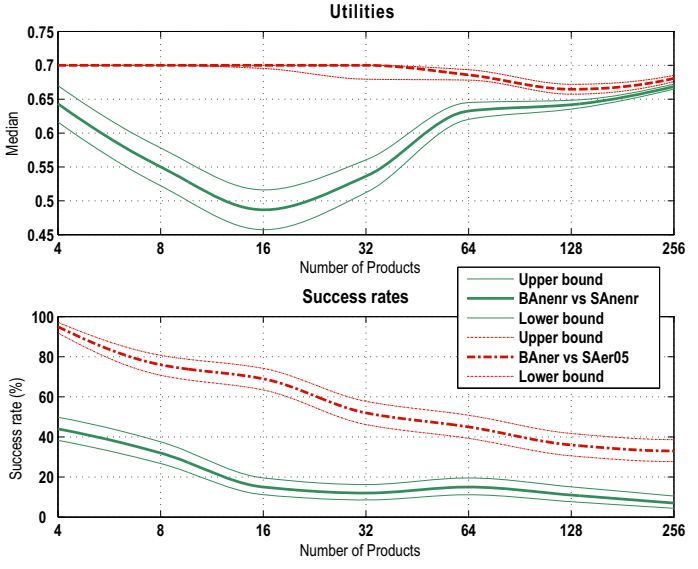


Fig. 1. Comparative of the BAnenr vs SAnenr and the BAner vs SAer05 strategies

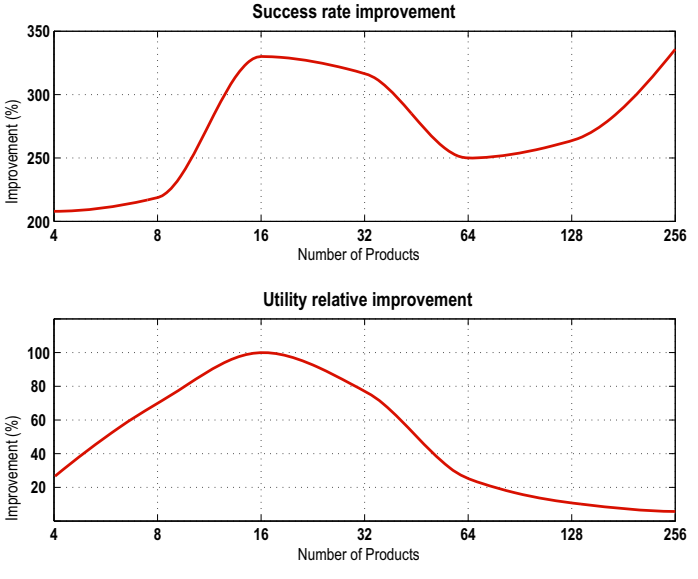


Fig. 2. Improvement in the success rate and relative improvement of utility

Finally, in Figure 2 two graphs are presented that depict the percentage improvement in the success rates and the comparative percentage improvement in utility. The improvement in the success rates portrays the comparative between the percentage of success rates obtained with the BAner vs SAer05 strategies

and those obtained with the BAnenr vs Sanenr strategies. This graph presents a very important property, which is the exponential trend of comparative improvements in the success rates. For catalogues with a small solution set the improvement is of approximately 200%, with an increase of around 325% for medium sized catalogues of 16 and 32 products being observed. It should be taken into account that when there are very few products, the possibility of a good solution being found at random, is greater than when the catalogue is large, which is why the improvement is smaller for 4 and 8 products. Although for 64 products the success rate decreases to 250%, in general, as the size of the catalogues increases there is an exponential tendency for the rates to improve. As the catalogues become very large, the probability of obtaining an optimal solution without expressivity decreases exponentially down to zero, whereas with expressivity the optimal solution is explicitly searched for.

The relative improvement in utility is a comparative measure that compares the improvements obtained with the BAner vs SAer05 strategies with respect the maximum improvement obtained. It can be observed that the reference catalogue is the one with 16 products, which is the scenario with which the maximum utility is obtained. So, the graph shows a percentage of relative improvement of 100% for this catalogue. For large catalogues the percentage of relative improvement decreases to below 10%. The minimum percentage improvement for smaller catalogues is 10% with an average value of around 60%.

4 Conclusions

This paper presents an experimental analysis of a fuzzy constraint based model for automated purchase negotiations in competitive trading environments. The analysis shows that the expressivity of the seller agent is essential to obtain an improvement in the negotiations. However, the expressivity of the buyer agent makes the results come close to those achieved with the reference non-expressive and non-receptive strategies. The viability of the potential sale offers is decisive in the improvement of the negotiations, and so, the **BAner vs SAenr** strategies, which concentrate on the seller agents' utility are invalid. To sum up, we can affirm that the expressivity factor brings significant benefit to the negotiation process and the key element resides in the expressivity of the seller agent and the receptivity of the buyer agent. However, it must be pointed out that under our negotiation model, an 'inexpressive' buyer agent is more expressive than an 'inexpressive' seller agent because a buyer agent expresses offers as a set of constraints, while a seller agent expresses offers as concrete products or rejections to purchase requirements.

As future work we propose the refinement of the mechanisms related to the purchase requirement valuation. As we have seen, the results using valuations of purchase requirements are not good because valuations distract the seller agent. We suggest to test different estimates for the viability parameter in the *prefer* function of the *generate potential sale offers* seller's mechanism. We believe that

including valuations in the purchase requirements the negotiation processes may be improved.

References

1. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter. MIT Press, Cambridge MA, USA (1994)
2. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems 24(3-4), 159–182 (1998)
3. Li, C., Giampapa, J.A., Sycara, K.: A review of research literature on bilateral negotiations. Technical Report CMU-RI-TR-03-41, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA (2003)
4. Lopez-Carmona, M.A., Velasco, J.R.: An expressive approach to fuzzy constraint based agent purchase negotiation. In: AAMAS-2006. Proceedings of the International Joint Conference on Autonomous Agents and Multi-agent Systems, Hakodate, Japan, pp. 429–431 (2006)
5. Lopez-Carmona, M.A., Velasco, J.R.: A fuzzy constraint based model for automated purchase negotiations. In: TADA/AMEC 2006. LNCS (LNAI), vol. 4452, pp. 234–247. Springer, Heidelberg (2007)
6. Luo, X., Jennings, N.R., Shadbolt, N., Ho-Fung-Leung, Lee, J.H.M.: A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. Artificial Intelligence 148(1-2), 53–102 (2003)
7. Dubois, D., Fargier, H., Prade, H.: Propagation and satisfaction of flexible constraints. Fuzzy Sets, Neural Networks and Soft Computing, pp. 166–187 (1994)
8. McBurney, P., Euk, R.M.V., Parsons, S., Amgoud, L.: A dialogue game protocol for agent purchase negotiations. Journal of Autonomous Agents and Multi-Agent Systems 7(3), 235–273 (2003)
9. Lai, R., Lin, M.W.: Modeling agent negotiation via fuzzy constraints in e-business. Computational Intelligence 20(4), 624–642 (2004)

Towards a Model Driven Process for Multi-Agent System

Tarek Jarraya[1] and Zahia Guessoum[2]

[1] I3S RAINBOW Team, University of Nice
Poly'tech Nice, 930 route des Colles BP145 06903 Sophia Antipolis, France
jarraya@i3s.unice.fr
[2] LIP6 OASIS Team, Univ. of Pierre and Marie Curie
104 avenue du Prsident Kennedy 75016 Paris, France
zahia.guessoum@lip6.fr

Abstract. We propose a new multi-agent development method, named MDAD (*Model Driven Agent Development*). It is based on the MDA (*Model Driven Architecture*) paradigm. The aim of MDAD method is to reduce the cost of building MAS applications by starting from abstract specification of system thanks to MAS meta-models, and producing the final system by means of transformations of this specification into computational entities. We present in this paper the application of MDAD to the INAF framework. First we give an overview of MDA approach and its application to MAS. Thus, several abstraction levels are determined and a set of meta-models is introduced. Then, we give the transformation rules used to produce INAF compliant models. MDAD method is illustrated with the timetable management benchmark.

1 Introduction

In the present state of research and development in the multi-agent area, we find contributions on development tools for multi-agent implementation [10] [1], on organizational models [8], and on methodologies for agent-oriented analysis and design [4]. However, most existing methodologies do not provide a complete development process (analysis, design, implementation, deployment...). Moreover, they do not rely on existing agent architectures and existing agent-based development tools. Recent research introduces meta-models to represent the main concepts and relationships of the proposed MAS methodologies. Moreover, an unification of these meta-models has been proposed by the AgentLink TFG AOSE [2].

This paper aims to bridge the gap between existing agent architectures and development tools. Our idea is to start from existing architectures and tools and to elaborate meta-models in order to formulate the necessary knowledge about the development process. Thus, unlike most existing methodologies which are based on a top-down approach, our design is bottom up. This new approach is inspired by the Model-Driven Architecture (MDA), proposed by OMG, which aims at separating application logic from the underlying technologies to improve

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 256–265, 2007.

reusability and development process [7]. The idea is that business knowledge should be permanent, whereas technical concerns are generally short-lived and limited to a given technology.

This paper shows the use of MDA to define a new MAS development process named MDAD which is based on a library of agent and organization meta-models. These meta-models may be used by the designer to easily describe the multi-agent models which are automatically transformed to generate an operational MAS. To illustrate our approach we present in this paper the different steps and models needed to develop the timetable management benchmark [3]. This application corresponds to a distributed MAS that helps at defining the timetable of professors and student groups. Each user has a personal assistant agent which manages his timetable. This agent interact with other agents to reserve a time slots for needed sessions. We present thus models introduced by developer and the ones generated by MDAD process.

This paper is organized as follow: The section 2 presents MDAD method. The section 3 presents the different meta-models used to represent the MAS independently from the platform level. The section 4 describes the meta-model of the INAF framework. The section 5 gives the required transformation rules between the two previous levels. The section 6 gives the generation code rules applied to PSM to obtain the MAS code.

2 MDAD: Model Driven Agent Development

The main idea of our approach is to reduce the cost of building a MAS application by starting from abstract specification of the MAS resulting from the methodology designing level, and generating the final system by means of transformations of this specification into computational entities.

MDAD aims to provide a set of meta-models and a set of knowledge-based systems that represent the transformation rules. These meta-models are used to describe models which can be considered as an operational specification of the system and as such they constitute the useful basic stones for the building of an operational MAS. The latter may be obtained by a succession of transformations which can be done automatically or semi-automatically. Some transformations require therefore some preliminary inputs from the designer. For instance, the agent meta-models are often sufficient to generate operational agents. However, the MAS deployment is seldom automatic.

2.1 Meta-models

Four abstract levels have thus been identified, from the most platform-specific to the most abstract:

– Infrastructure level: represents the components of the MAS at the implementation level (code); it includes the domain implementation, the generated agent implementation and the MAS deployment,

- PSM (platform-specific model) level: represents the components of the MAS from the platform point of view. This representation makes abstraction of some implementation details (language, middleware...). It includes the description of agent classes, meta behaviors building, MAS initialization classes and domain classes diagram,
- PIM (platform-independent model) level: represents the requirements of the MAS at design time, independently of the underlying platforms; MAS organization and meta-behaviors descriptions belong to this level,
- CIM (computation-independent model) level: represents the statements of a problem in a given domain;

In MDAD, we use UML diagrams to describe PIM and PSM. To represent our meta-models, we have defined MAS profiles. We thus have defined two profiles to represent PIM and PSM meta-models.

2.2 MDAD Development Process

The PIM is an abstract representation of the MAS application including mainly the specification of the MAS organization and agents behavior. The PSM describes the components of the MAS from the platform point of view. This representation makes abstraction of some implementation details (language, middleware...).

We have defined a set of transformation rules ensuring the crossing from PIM to PSM. These rules express the know-how of a developer knowing the structure of the MAS obtained starting from methodology, as well as, the structure and the services offered by the implementation platform. These rules are written in the QVT language [9]. The obtained PSM by application of transformation rules will be used to generate MAS code. This code contains agent classes, MAS initialization classes, and domain classes.

Thus, the developer needs the following steps to implement its MAS:

1. Defining the PIM by using MAS Profile,
2. Applying transformation rules to the PIM to produce the PSM,
3. Applying generation code rules to the PSM to obtain the application code.

Our approach is enough generic, it can be applied to different MAS methodologies and platforms. We present in this paper the application of our approach to a MAS abstract level defined by a set of meta-models and its implementation with INAF framework [11]. The paper gives some meta-models for PIM and PSM, and analyzes their transformation. The aim is to show that MDAD facilitates the development of MAS.

3 Platform Independent Level

Several methodologies have been proposed to develop MAS which propose different approaches and models to facilitate this development. The design of MAS requires often the definition of a domain model, an organizational model,

and one or several agent models. Each existing methodology relies on only one meta-model. The whole MAS is thus defined as an instance of this meta-model.

However, Demazeau [6] propose in the Vowels methodology to use a set of models to define the multi-agent system. These models concern agent, interaction, organization and environment aspects. Thus, several meta-models can be therefore introduced to represent each aspect, and the MAS will be defined as a combination of these models.

Our approach relies on this idea. We propose a library of meta-models for domain, organizational, and agent aspects. The main advantage of this approach is to provide more flexibility in the specification of the MAS. Thus, the designer can use the more appropriate agent meta-model (for instance the BDI agent) to describe the agents of its MAS application. Being limited by the number of page, we describe in this paper only an example of agent meta-model.

Our agent meta-model presents a basic agent representation which is similar to the meta-model given in [5]. The *Agent* concept has one *Goal* and executes a set of actions. The agent is defined as *Class* stereotype, and its behavior can be described by an *Activity* instance, which is a sequence of actions.

Concerning the *Goal* concept, we define it as a stereotype of *Constraint* meta-class. Thus, *Goal* is defined as a boolean expression that the agent can evaluate during its activity. This expression is written with OCL language.

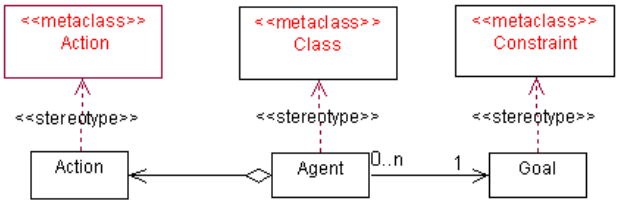
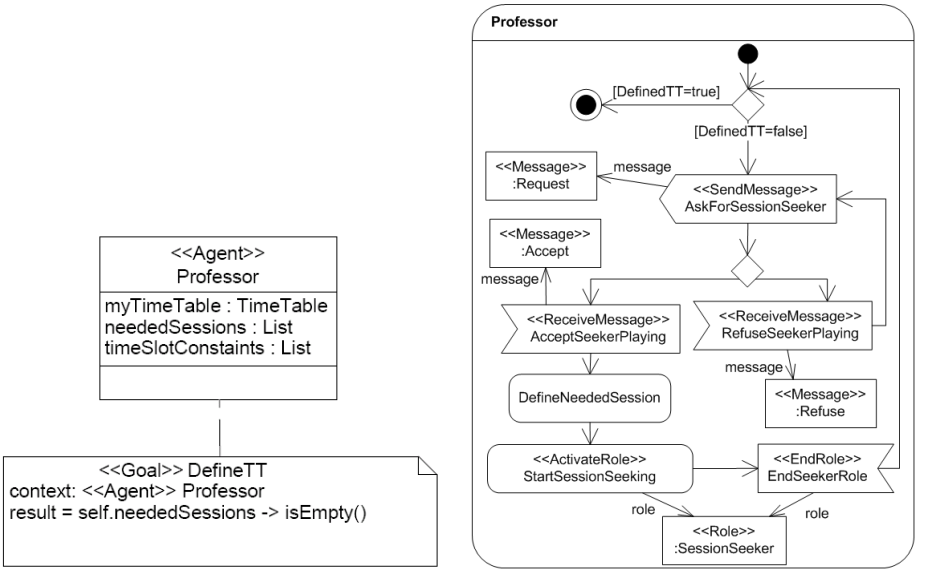


Fig. 1. Agent meta-model

At the model level, the developer uses class diagrams to describe the agent structure, and activity diagrams to represent its behavior. Figure 2a shows the *Professor* agent structure. Its goal is to define the meetings of its timetable. Since the attribute *neededSession* is a list which contains the meetings to to define, so we can express the *Professor Goal* by a boolean expression: the list *neededSessions* is empty.

The *Professor* agent behavior is represented by an activity diagram, see figure 2b. The agent starts by evaluating its *Goal*. If its goal is not yet reached, then it asks the *TimeTableManaging* group manager to play the *SessionSeeker* role, by means of the action *RequestRole*. If the request is accepted, then it defines the needed meeting attributes. Next it activates the *SessionSeeker* role. At the end of the role, the agent evaluates again its *Goal*.



(a) *Professor* agent class (b) *Professor* agent activity

Fig. 2. *Professor* agent

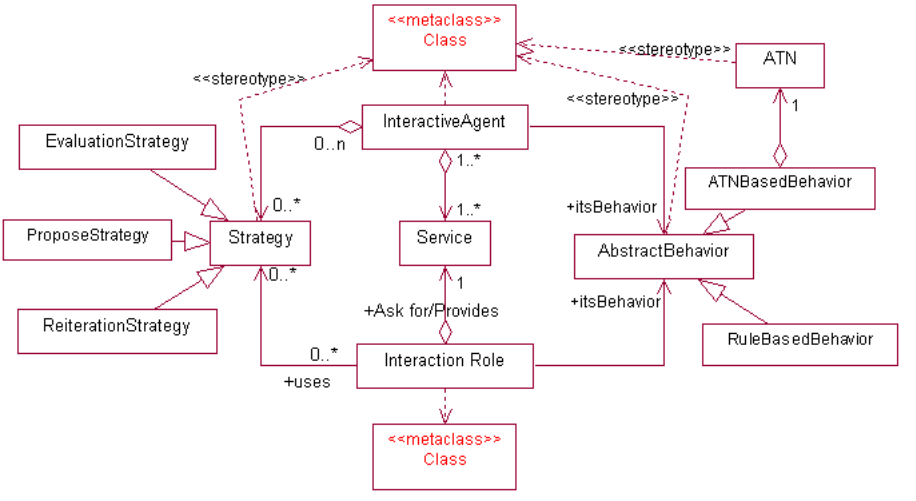


Fig. 3. INAF meta-model

4 Platform Specific Level

The implementation of a MAS involves the implementation of various agents by choosing the suitable agent architectures. This implementation requires deep technical knowledge (component architecture, programming language, middleware...). A first abstraction level allows therefore hiding some technical details that can be automatically deduced to simplify the implementation process. For instance, the agent engine is often generic. It is not therefore useful to have details about this engine to implement the agent.

We describe in this section the INAF framework meta-model [11]. The agent architecture model of INAF may be seen as an open model that can be then enriched by incremental refinement. The agent behavior can be implemented by an ATN or a rule base. For agents interaction, INAF implement FIPA interaction protocols. The *Interaction Role* concept is part of the protocol activity, which can be played by one or more agents during an interaction. The *Strategy* concept is a decisional action which indicates which action to undertake while being based on the goals and knowledge of the agent. In INAF, the *Service* concept corresponds to the interaction object. Any interaction relates to one service whose providing indicates the interaction success or failure.

5 Transformation Process

We present in this section the transformation rules needed to transit from an agent model to an implementation model compliant to the INAF framework.

5.1 Transformation Rules

We give in this section the main transformation rules necessary to implement the *Agent* concept with INAF framework. So, we start by writing the *Agent* concept transformation rules.

Rule 1: For an *Agent* x, if all its roles were transformed (i.e. the agent played roles are given in the the organization model), and if its behavior is described by an *Activity* g, then:

1. We create an *InteractiveAgent* y with the name and attributes of x.
2. Then, we invoke the rule 4 to transform g into an *ATN* t.
3. Lastly, we attribute t to the attribute *behavior* of y.

Rule 2: For a *Goal* g which is associated to a *Role* r, and r is already transformed into an *ATNBasedBehavior* or an *AbstractBehavior* c, then we add to c a boolean *Operation* o with the name of g. The body of this operation is the g OCL body expression.

Rule 3: For a *Goal* g which is associated to an *Agent* a and that a is already transformed into one *InteractiveAgent* i, then we add to i a boolean *Operation* o named *"isActive"*. The body of o is the g OCL body expression.

Rule 4: For an *Activity* g then:

1. We create an *ATN* *t* with the same name of *g*.
2. Then an interpreter is used to cross *g*. For each activity element we call the corresponding transformation rule.

5.2 Timetable PSM

At the beginning of transformation, *rule 1* is the only active rule. So, rule 3 is applied twice to produce two interactive agents: *Professor* and *StudentGroup*. Figure 4 shows the *Professor* class. The transformation of its *Goal DefineTT* generates a boolean operation *defineTT()* whose body takes the *DefineTT* OCL expression. We keep at this transformation level the OCL notation, because PSM concepts associated to these expressions, are not yet transformed to Java code.

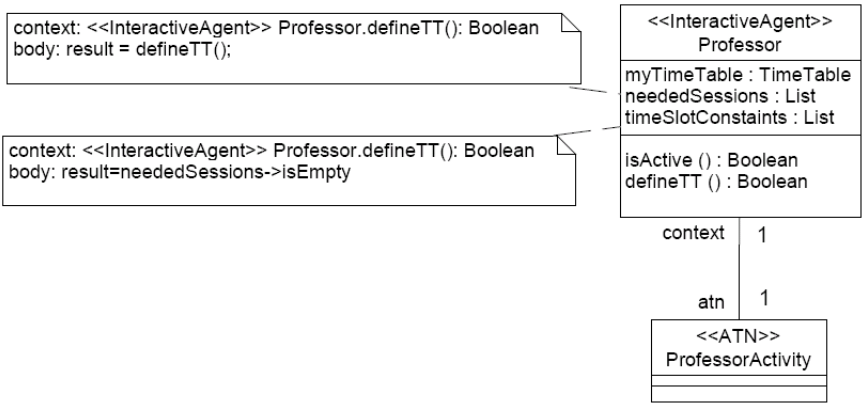


Fig. 4. *Professor* interactive agent

The invocation of rule 4 generate two *ATN* of *Professor* and *StudentGroup* agents. *SendMessage* actions are transformed into actions. *ReceiveMessage* actions are transformed into conditions. The condition takes again the name of the action *ReceiveMessage*. The message related to an action *SendMessage* or *ReceiveMessage* is taken again on level PSM. Figure 5 shows the model elements (i.e. *State*, *Condition* and *Transition*), which compose *ProfessorBehavior*.

The know-how needed to transit from the PIM to a PSM compliant to INAF, can be summarized in these points:

1. The agent behavior is implemented by *ATN*.
2. The *Goal* concept does not have a correspondence in INAF. We propose to implement it by a boolean operation associated to the agent class.

The PSM obtained after rules execution can be refined in the aim to improving it and bring it more closer to the operational system.

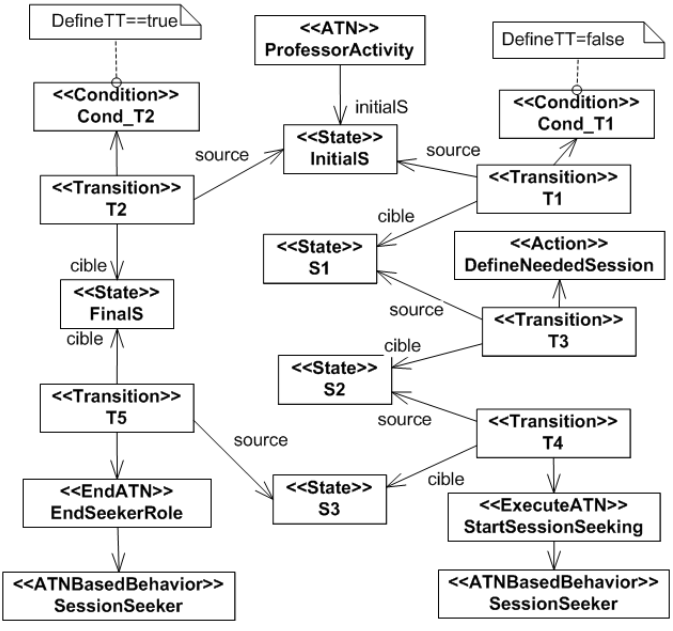


Fig. 5. *ProfessorBehavior* ATN class

6 Code Generation

The third stage of our method *MDAD* is to generation platform code, starting from PSM. To obtain an operational MAS we must generate the application classes which are platform complaint. These classes are an extension of the platform basic classes. We give in what follows rules allowing the generation of INAF classes. This intermediate model is important for a clear representation the system structure, Moreover it is more easier to understand than INAF code.

6.1 Generation Rules

Rule 1: For an *InteractiveAgent* A, we add a new Class c which extends *InteractiveAgent* Class.

Rule 2: For an *ATN* T, and if *Agent* A which is associated to T, was transformed into a Class C, then:

1. we add an operation *buildATN()* to the class C.
2. we add the code line : " ATN atn = new ATN();"
3. we cross T starting from its initial state
4. For each *Transition*:
   - For each *Condition* invoke rule 3
   - For each *Action* invoke rule 4

Rule 3: For a *Condition* C of an *ATN* T then:

1. We generate code line " NamedCondition " + *C.name*+" = new NamedCondition("+C.name+");".
2. We add a boolean operation *O* of the name *C.name* with the class *A* which corresponds to the attribute *T.context*.
3. If there is a *Message M* associated with *Condition C*, then we add to *O* the code line: " return currentMessage.getPerforamtive()="+ *M.name*+";"

Rule 4: For an *Action A* of an *ATN T* then:

1. We generate the code line " NamedAction " + *A.name*+" = new NamedAction("+ *A.name*+");".
2. We add an operation *O* with the name *A.name* to *C* which corresponds to *T.context* attribute.

6.2 Timetable Code

The application of the rules described previously on model PSM enables us to generate system classes which are specialization of INAF basic classes. This step allows to build the *InteractiveAgent* classes by application of rule 1.

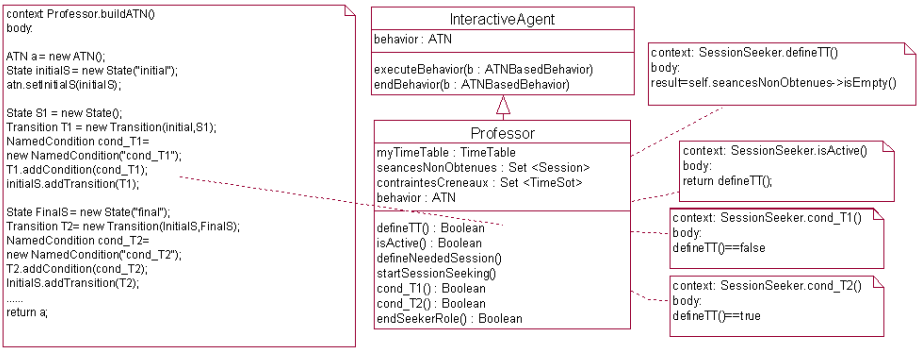


Fig. 6. *Professor* class

Figure 6 describes the *Professor* class which extends the *InteractiveAgent* class. It implements the static *buildATN()* method. The Java code of this method is given in a note associated to *Professor* class. This code is obtained by application of rule 2. The methods *defineNeededSession* and *startSessionSeeking* are obtained by application of rule 4 to the all actions of *ProfessorActivity ATN*. The boolean methods *cond_T1*, *cond_T2* and *endSeekerRole* are obtained by application of rule 3 to the conditions of *ProfessorActivity ATN*.

7 Conclusion

This paper presented a new approach, named MDAD, for agent oriented software engineering. MDAD is based on the MDA approach. It is founded on existing

agent technologies and theories. Several meta-models and the needed knowledge to transform them in an operational MAS have been introduced.

A first implementation of our method is done using the DIMA platform [10] and the MetaGen development tool [12]. MetaGen has been used to define meta-models. Models Editors are automatically generated from meta-models description. These editors are used by the designer to describe models. DIMA is therefore used to run and deploy the generated code.

The first development with MDAD showed that this new method is interesting and promising; it improves the multi-agent development process. However, more experiments with real-life applications are needed to validate the proposed approach. We are therefore currently working on the development of more a complex example in order to better evaluate our method.

References

1. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade Programmer's guide JADE 3.2 (July 2004)
2. Bernon, C., Cossentino, M., Gleizes, M.P., Turci, P., Zambonelli, F.: A study of some multi-agent meta-models. In: Agent Oriented Software Engineering Workshop helt at AAMAS'04, pp. 62–77 (2004)
3. Bernon, C., Gleizes, M.-P., Glize, P., Picard, G.: Problme de l'emploi du temps. Technical report, Groupe de travail ASA (2003)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
5. Cossentino, M., Bernon, C., Pavon, J.: Modelling and meta-modelling issues in agent oriented software engineering: The agentlink aose tfg approach. Technical Report Agent Link III, The AgentLink AOSE Technical Forum Group, Ljubljana, Slovenia (February 28, 2005)
6. Demazeau, Y.: Vowels. In: IWDAIMAS'96. Invited lecture, 1st Ibero-American Workshop on Distributed AI and Multi-Agent Systems, Mexico (1996)
7. OMG TC Document: Model driven architecture (mda). Technical report, OMG (2001)
8. Ferber, J., Gutknecht, O.: Aalaadin: a meta-model for the analysis and design of organizations in mutli-agent systems. In: Demazeau, Y. (ed.) ICMAS'98, Paris, pp. 128–135 (1998)
9. Object Managment Group: Mof qvt final adopted specification. Technical Report ptc/05-11-01, OMG (Novembre 2005)
10. Guessoum, Z., Briot, J.-P.: From active objects to autonomous agents. IEEE Concurrency 7(3), 68–76 (1999)
11. Jarraya, T., Guessoum, Z.: Reuse interaction protocols to develop interactive agents. In: Liu, J., Wah, B.W. (eds.) Proceedings of IAT conference, Hong Kong, China, Dcembre 18-22, 2006, IEEE Press, Los Alamitos (2006)
12. Revault, N., Sahraoui, H., Blain, G., Perrot, J.-F.: A metamodeling technique: The mtagen system. In: Beilner, H., Bause, F. (eds.) MMB 1995 and TOOLS 1995. LNCS, vol. 977, pp. 127–139. Springer, Heidelberg (1995)

Towards an Epistemic Logic for Uncertain Agents*

Zining Cao

Department of Computer Science and Technology
Nanjing University of Aero. and Astro., Nanjing 210016, P.R. China
caozn@nuaa.edu.cn

Abstract. In this paper, we propose a logic for reasoning about probabilistic belief, called PBL_r. Our language introduces formulas that express "agent i believes that the probability of φ is at least p". We first provide an inference system of PBL_r, and then introduce a probabilistic semantics for PBL_r. The soundness and finite model property of PBL_r are proved, which ensure the weak completeness and decidability of PBL_r.

1 Introduction

The study of knowledge and belief has a long tradition in philosophy. More recently, researchers in artificial intelligence and theoretical computer science have become increasingly interested in reasoning about knowledge and belief. In wide areas of application of reasoning about knowledge and belief, it is necessary to reason about uncertain information. Therefore the representation and reasoning of probabilistic information in belief is important.There are a lot of literatures related to the representation and reasoning of probabilistic information, such as probabilistic knowledge logic [4], probabilistic dynamic epistemic logic [8], probabilistic temporal epistemic logic [3] and etc.

A distinguished work was done by Fagin and Halpern [4], in which a probabilistic knowledge logic was proposed. It expanded the language of knowledge logic by adding formulas like "$w_i(\varphi) \geq 2w_i(\psi)$" and "$w_i(\varphi) < 1/3$", where φ and ψ are arbitrary formulas. These formulas mean "φ is at least twice probable as ψ" and "φ has probability less than 1/3". The typical formulas of their logic are "$a_1 w_i(\varphi_1) + ... + a_k w_i(\varphi_k) \geq b$", "$K_i(\varphi)$" and "$K_i^b(\varphi)$", the latter formula is an abbreviation of "$K_i(w_i(\varphi) \geq b)$". Here formulas may contain nested occurrences of the modal operators w_i and K_i, and the formulas in [5] do not contain nested occurrences of the modal operators w_i. On the basis of knowledge logic, they added axioms of reasoning about linear inequalities and probabilities. To provide semantics for such logic, Fagin and Halpern introduced a probability space on Kripke models of knowledge logic, and gave some conditions about probability space, such as OBJ, SDP and $UNIF$. At last, Fagin and Halpern

* This work was supported by the National Natural Science Foundationof China under Grant 60473036.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 266–276, 2007.

concluded by proving the soundness and weak completeness of their probabilistic knowledge logic.

Kooi's work [8] combined the probabilistic epistemic logic with the dynamic logic yielding a new logic, $PDEL$, that deals with changing probabilities and takes higher-order information into account. The syntax of $PDEL$ is an expansion of Fagin and Halpern's logic by introducing formula "$[\varphi_1]\varphi_2$", which can be read as "φ_2 is the case, after everyone simultaneously and commonly learns that φ_1 is the case". The semantics of $PDEL$ is the same as Fagin and Halpern's semantics, which is based on a combination of Kripke structure and probability functions. The soundness and weak completeness of $PDEL$ was proved.

In [9], the authors presented a probabilistic belief logic, called PEL, which is essentially a restricted version of the logic proposed by Fagin and Halpern. But the inference system was not given and the corresponding properties such as soundness and completeness of PEL were not studied.

In [7], Hoek investigated a probabilistic logic P_FD. This logic was enriched with operators $P_r^>, (r \in [0,1])$ where the intended meaning of $P_r^>\varphi$ is "the probability of φ is strictly greater than r". The author gave a completeness proof of P_FD by the construction of a canonical model for P_FD considerably. Furthermore, the author also proved finite model property of the logic by giving a filtration-technique for the intended models. Finally, the author proved the decidability of the logic. The logic P_FD is based on a set F, where F is a finite set and $\{0,1\} \subseteq F \subseteq [0,1]$. The completeness of P_FD was not proved in [7] for the case that F is infinite. Hoek presented this problem as an open question and considered it as a difficult task. He thought this problem may be tackled by introducing infinitary rules.

In [3], a temporal epistemic logic was presented, which is essential a combinational logic of P_FD and temporal logic. This logic was used to represent and verify some properties in multi-agent systems.

In this paper, we propose a probabilistic belief logic. There is no axiom and rule about linear inequalities and probabilities in the inference system of probabilistic belief logic. Hence the inference system looks simpler than Fagin and Halpern's logic. We also propose a simpler semantics for probabilistic belief logic, where is no accessible relation and can be generalized to description semantics of other probabilistic modal logics. Moreover, we present a new proof for the finite model property of our probabilistic belief logic. This logic can be used for the specification of uncertain agents. To verify the correctness of uncertain multi-agent systems, one can develop a model checking algorithm for PBL_r, which can be used to check whether uncertain agents satisfy the given PBL_r property.

The remainder of the paper is organized as follows: In Section 2, we propose a probabilistic belief logic, called PBL_r. PBL_r has a syntax restriction that the probability a in the scope of $B_i(a, \varphi)$ must be a rational number. Here subscript r in PBL_r means rational numbers. We provide the probabilistic semantics of PBL_r, and prove the soundness and finite model property of PBL_r. From the finite model property, we obtain the weak completeness of PBL_r. Furthermore, the decidability of PBL_r is shown. The conclusion is given in Section 3.

2 PBL_r and Its Probabilistic Semantics

There are examples of probabilistic belief in daily life. For example, one may believe that the probability of "it will rain tomorrow" is less than 0.4. In order to represent and reason with probabilistic belief, it is necessary to extend belief logic to probabilistic belief logic. In following, we propose a probabilistic belief logic PBL_r, the basic formula in PBL_r is $B_i(a, \varphi)$, which says agent i believes that the probability of φ is no less than a. In the semantics of PBL_r, we assign an inner probability space to every possible world in the model, here "inner" means the measure does not obey the additivity property, but obeys some weak additivity properties of inner probability measure.

2.1 Language of PBL_r

Throughout this paper, we let L^{PBL_r} be a language which is just the set of formulas of interest to us.

Definition 1. The set of well formed formulas in PBL_r, called L^{PBL_r}, is given by the following rules:

(1) If $\varphi \in$Atomic formulas set *Prop*, then $\varphi \in L^{PBL_r}$;

(2) If $\varphi \in L^{PBL_r}$, then $\neg\varphi \in L^{PBL_r}$;

(3) If φ_1,$\varphi_2 \in L^{PBL_r}$, then $\varphi_1 \wedge \varphi_2 \in L^{PBL_r}$;

(4) If $\varphi \in L^{PBL_r}$ and a is a rational number in $[0, 1]$, then $B_i(a, \varphi) \in L^{PBL_r}$, where i belongs to the set of agents $\{1, ..., n\}$. Intuitively, $B_i(a, \varphi)$ means that agent i believes the probability of φ is no less than a.

2.2 Semantics of PBL_r

We will describe the semantics of PBL_r, that is, a formal model that we can use to determine whether a given formula is true or false. We call the formal model inner probabilistic model, roughly speaking, at each state, each agent has an inner probability on a certain set of states.

Definition 2. An inner probabilistic model PM of PBL_r is a tuple $(S, P_1, ..., P_n, \pi)$, where

(1) S is a nonempty finite set whose elements are called possible worlds or states.

(2) P_i is a mapping, it maps every possible world s to a PBL_r-probability space $P_i(s) = (S, X, \mu_{i,s})$, here $X = \wp(S)$;

$\mu_{i,s}$ is a PBL_r-inner probability measure that assigns a rational number to every element of X, which means that $\mu_{i,s}$ satisfies the following conditions:

(a) $0 \leq \mu_{i,s}(A) \leq 1$ for all $A \in X$;

(b) $\mu_{i,s}(\emptyset) = 0$ and $\mu_{i,s}(S) = 1$;

(c) If $A_1, A_2 \in X$ and $A_1 \subseteq A_2$, then $\mu_{i,s}(A_1) \leq \mu_{i,s}(A_2)$;

(d) If $A_1, A_2 \in X$ and $A_1 \cap A_2 = \emptyset$, then $\mu_{i,s}(A_1 \cup A_2) \geq \mu_{i,s}(A_1) + \mu_{i,s}(A_2)$;

(e) If $A_1, A_2 \in X$, then $\mu_{i,s}(A_1 \cap A_2) \geq \mu_{i,s}(A_1) + \mu_{i,s}(A_2) - 1$;

(f) Let $\Lambda_{i,s} = \{s' | P_i(s) = P_i(s')\}$, then $\mu_{i,s}(\Lambda_{i,s}) = 1$.

(3) π is a mapping: $S \times Prop \rightarrow \{true, false\}$, where $Prop$ is an atomic formulas set.

Remark: It is easy to see that the conditions (d) and (e) in Definition 2 are weaker than the finite additivity condition in probability measure. One can check that if μ is a probability measure, then inner measure μ^* (here $\mu^*(A) = sup(\{\mu(B)|B \subseteq A\})$) induced by μ obeys the conditions (d) and (e) in Definition 2, i.e., the reason we call $\mu_{i,s}$ inner probability measure.

The notation $\Lambda_{i,s}$ in the condition (f) represents the set of states whose probability space is same as the probability space of state s. Therefore the condition (f) means that for any state s, the probability space of almost all states is the same as the probability space of s.

Definition 3. Probabilistic semantics of PBL_r

$(PM, s) \models p$ iff $\pi(s, p) = true$, where p is an atomic formula;

$(PM, s) \models \neg\varphi$ iff $(PM, s) \not\models \varphi$;

$(PM, s) \models \varphi_1 \wedge \varphi_2$ iff $(PM, s) \models \varphi_1$ and $(PM, s) \models \varphi_2$;

$(PM, s) \models B_i(a, \varphi)$ iff $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$, where $ev_{PM}(\varphi) = \{s'|(PM, s') \models \varphi\}$.

2.3 Inference System of PBL_r

Now we list a number of valid properties of probabilistic belief, which form the inference system of PBL_r.

Axioms and inference rules of proposition logic

Axiom 1. $B_i(0, \varphi)$ (For any proposition φ, agent i believes that the probability of φ is no less than 0.)

Axiom 2. $B_i(a, \varphi) \wedge B_i(b, \psi) \rightarrow B_i(max(a+b-1, 0), \varphi \wedge \psi)$ (For any φ and ψ, if agent i believes that the probability of φ is no less than a, and believes that the probability of ψ is no less than b, then agent i believes that the probability of $\varphi \wedge \psi$ is no less than $max(a + b - 1, 0)$.)

Axiom 3. $B_i(a, \varphi) \rightarrow B_i(1, B_i(a, \varphi))$ (If agent i believes that the probability of φ is no less than a, then agent i believes that the probability of his belief being true is no less than 1.)

Axiom 4. $\neg B_i(a, \varphi) \rightarrow B_i(1, \neg B_i(a, \varphi))$ (If agent i believes that the probability of φ is less than a, then agent i believes that the probability of his belief being true is no less than 1.)

Axiom 5. $B_i(a, \varphi) \rightarrow B_i(b, \varphi)$, where $1 \geq a \geq b \geq 0$. (If agent i believes that the probability of φ is no less than a, and $1 \geq a \geq b \geq 0$, then agent i believes that the probability of φ is no less than b.)

Rule 1. $\vdash \varphi \Rrightarrow \vdash B_i(1, \varphi)$ (If φ is a tautology proposition, then agent i believes that the probability of φ is no less than 1.)

Rule 2. $\vdash \varphi \rightarrow \psi \Rrightarrow \vdash B_i(a, \varphi) \rightarrow B_i(a, \psi)$ (If $\varphi \rightarrow \psi$ is a tautology proposition, and agent i believes that the probability of φ is no less than a, then agent i believes that the probability of ψ is no less than a.)

Rule 3. $\vdash \neg(\varphi \wedge \psi) \Rrightarrow \vdash \neg(B_i(a, \varphi) \wedge B_i(b, \psi))$ for any a, $b \in [0, 1]$ such that $a + b > 1$. (If φ and ψ are incompatible propositions, then it is impossible that

agent i believes that the probability of φ is no less than a, and believes that the probability of ψ is no less than b, where $a + b > 1$.)

Rule 4. $\vdash \neg(\varphi \wedge \psi) \Rightarrow \vdash B_i(a, \varphi) \wedge B_i(b, \psi) \rightarrow B_i(a+b, \varphi \vee \psi)$, where $a+b \leq 1$. (If φ and ψ are incompatible propositions, agent i believes that the probability of φ is no less than a, and believes that the probability of ψ is no less than b, where $a + b \leq 1$, then agent i believes that the probability of $\varphi \vee \psi$ is no less than $a + b$.)

Remark: It is necessary to note that by the definition of well formed formulas of PBL_r, all the probabilities in the axioms and inference rules of PBL_r should be rational numbers. For example, *Axiom* 5: "$B_i(a, \varphi) \rightarrow B_i(b, \varphi)$, where $1 \geq a \geq b \geq 0$", here a, b are rational numbers". Since the probabilities a and b in the formulas $B_i(a, \varphi)$ and $B_i(b, \psi)$ are rational numbers, so the probability $max(a + b-1, 0)$ in the scope of $B_i(max(a+b-1, 0), \varphi \wedge \psi)$ in *Axiom* 2 and the probability $a + b$ in the scope of $B_i(a + b, \varphi \vee \psi)$ in *Rule* 4 are also rational numbers.

We say φ is provable from Γ in PBL_r, and write $\Gamma \vdash_{PBL_r} \varphi$, if there is a proof from Γ to φ in PBL_r.

2.4 Soundness of PBL_r

We will prove that PBL_r characterizes the set of formulas that are valid with respect to probabilistic model. Inference system of PBL_r is said to be sound with respect to probabilistic models if every formula provable in PBL_r is valid with respect to probabilistic models. The system PBL_r is complete with respect to probabilistic models if every formula valid with respect to probabilistic models is provable in PBL_r.

Now, we give the following proposition:

Proposition 1. (Soundness of PBL_r) If $\Gamma \vdash_{PBL_r} \varphi$, then $\Gamma \models_{PBL_r} \varphi$.

Proof. We show axiom and each rule of PBL_r is sound, respectively.

Axiom 1: By the definition of PBL_r-probability measure, for any s, if $A \in X$ then $\mu_{i,s}(A) \geq 0$. Since for any φ, $ev_{PM}(\varphi) = \{s' | (PM, s') \models \varphi\} \in X$, we have $\mu_{i,s}(ev_{PM}(\varphi)) \geq 0$, therefore $B_i(0, \varphi)$ holds.

Axiom 2: Suppose $(PM, s) \models B_i(a, \varphi) \wedge B_i(b, \psi)$, so $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$ and $\mu_{i,s}(ev_{PM}(\psi)) \geq b$. For $\mu_{i,s}$ is PBL_r-probability measure, we get $\mu_{i,s}(ev_{PM}(\varphi \wedge \psi)) = \mu_{i,s}(ev_{PM}(\varphi) \cap ev_{PM}(\psi)) \geq \mu_{i,s}(ev_{PM}(\varphi)) + \mu_{i,s}(ev_{PM}(\psi)) - 1 \geq a+b-1$, which implies $(PM, s) \models B_i(max(a + b - 1, 0), \varphi \wedge \psi)$.

Axiom 3: Suppose $(PM, s) \models B_i(a, \varphi)$, therefore $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$. Let $\Lambda_{i,s} = \{s' | P_i(s) = P_i(s')\}$, then $\Lambda_{i,s} \in X$ and $\mu_{i,s}(\Lambda_{i,s}) = 1$. Let $\Xi = \{s' | \mu_{i,s'}(ev_{PM}(\varphi)) \geq a\}$. Since $s' \in \Lambda_{i,s}$ implies $s' \in \Xi$, it is clear $\Lambda_{i,s} \subseteq \Xi$, since $\mu_{i,s}(\Lambda_{i,s}) = 1$, so $\mu_{i,s}(\Xi) = 1$. If $s' \in \Xi$, then $s' \in ev_{PM}(B_i(a, \varphi))$, therefore $\mu_{i,s}(ev_{PM}(B_i(a, \varphi))) = 1$, we get $(PM, s) \models B_i(1, B_i(a, \varphi))$ as desired.

Axiom 4: Suppose $(PM, s) \models \neg B_i(a, \varphi)$, so $\mu_{i,s}(ev_{PM}(\varphi)) < a$. Let $\Lambda_{i,s} = \{s' | P_i(s) = P_i(s')\}$, then $\Lambda_{i,s} \in X$ and $\mu_{i,s}(\Lambda_{i,s}) = 1$. Let $\Xi = \{s' | \mu_{i,s'}(ev_{PM}(\varphi)) < a\}$, for $s' \in \Lambda_{i,s}$ implies $s' \in \Xi$, it is clear $\Lambda_{i,s} \subseteq \Xi$, since $\mu_{i,s}(\Lambda_{i,s}) = 1$, so $\mu_{i,s}(\Xi) = 1$. If $s' \in \Xi$, then $s' \in ev_{PM}(B_i(a, \varphi))$, therefore $\mu_{i,s}(ev_{PM}(\neg B_i(a, \varphi))) = 1$, we get $(PM, s) \models B_i(1, \neg B_i(a, \varphi))$ as desired.

Axiom 5: Suppose $(PM, s) \models B_i(a, \varphi)$, so $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$. If $1 \geq a \geq b \geq 0$, then $\mu_{i,s}(ev_{PM}(\varphi)) \geq b$, so $(PM, s) \models B_i(b, \varphi)$, therefore $B_i(a, \varphi) \rightarrow B_i(b, \varphi)$ holds.

Rule 1: Since $\models \varphi$, so for any possible world s, $\mu_{i,s}(ev_{PM}(\varphi)) \geq 1$, therefore $\models B_i(1, \varphi)$ holds.

Rule 2: Since $\models \varphi \rightarrow \psi$, so $ev_{PM}(\varphi) \subseteq ev_{PM}(\psi)$. Suppose $(PM, s) \models B_i(a, \varphi)$, therefore $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$, by the property of PBL_r-probability space, we get $\mu_{i,s}(ev_{PM}(\varphi)) \leq \mu_{i,s}(ev_{PM}(\psi))$. So $\mu_{i,s}(ev_{PM}(\psi)) \geq a$. Therefore $(PM, s)| = B_i(a, \psi)$, and *Rule* 2 of PBL_ω holds.

Rule 3: Suppose $\models \neg(\varphi \wedge \psi)$, so $ev_{PM}(\varphi) \cap ev_{PM}(\psi) = \emptyset$. By the property of PBL_r-probability space, for any possible world s, we get $\mu_{i,s}(ev_{PM}(\varphi) \cup ev_{PM}(\psi)) = \mu_{i,s}(ev_{PM}(\varphi)) + \mu_{i,s}(ev_{PM}(\psi))$ and $\mu_{i,s}(ev_{PM}(\varphi) \cup ev_{PM}(\psi)) \leq 1$, therefore $\mu_{i,s}(ev_{PM}(\varphi)) + \mu_{i,s}(ev_{PM}(\psi)) \leq 1$. Assume $(PM, s) \models (B_i(a, \varphi) \wedge B_i(b, \psi))$ where $a + b > 1$, then $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$, $\mu_{i,s}(ev_{PM}(\psi)) \geq b$, but $a + b > 1$, it is a contradiction.

Rule 4: Suppose $\models \neg(\varphi \wedge \psi)$ and for possible world s, $(PM, s) \models B_i(a, \varphi) \wedge B_i(b, \psi)$, so $ev_{PM}(\varphi) \cap ev_{PM}(\psi) = \emptyset$, $\mu_{i,s}(ev_{PM}(\varphi)) \geq a$, and $\mu_{i,s}(ev_{PM}(\psi)) \geq b$. By the property of PBL_r-probability space, for any possible world s, we get $\mu_{i,s}(ev_{PM}(\varphi) \cup ev_{PM}(\psi)) = \mu_{i,s}(ev_{PM}(\varphi)) + \mu_{i,s}(ev_{PM}(\psi))$. Hence, $\mu_{i,s}(ev_{PM}(\varphi)) + \mu_{i,s}(ev_{PM}(\psi)) \geq a + b$ and $\mu_{i,s}(ev_{PM}(\varphi) \cup ev_{PM}(\psi)) \geq a + b$, which means $(PM, s) \models B_i(a + b, \varphi \vee \psi)$.

2.5 Finite Model Property and Decidability of PBL_r

One may want to prove the completeness property of PBL_r. Unfortunately, this property does not hold, for example, $\{B_i(1/2, \varphi), B_i(2/3, \varphi), ..., B_i(n/n + 1, \varphi), ...\} \cup \{\neg B_i(1, \varphi)\}$ is not satisfied in any PBL_r-model, but any finite subset of it has a model. Therefore we have to seek for a weak version of completeness property, i.e., weak completeness. The weak completeness of PBL_r means that for any **finite** formulas set Γ and any formula φ, we have $\Gamma \models_{PBL_r} \varphi \Rightarrow \Gamma \vdash_{PBL_r} \varphi$.

In order to prove the weak completeness of PBL_r, we first present a probabilistic belief logic - $PBL_r(N)$, where N is a given natural number. The finite model property of $PBL_r(N)$ is then proved. From this property, we get the weak completeness and the decidability of PBL_r.

The syntax of $PBL_r(N)$ is the same as the syntax of PBL_r except that the probabilities in formulas should be rational numbers like k/N. For example, every probability in formulas of $PBL_r(3)$ should be one of 0/3, 1/3, 2/3 or 3/3. Therefore, $B_i(1/3, \varphi)$ and $B_i(2/3, B_j(1/3, \varphi))$ are well formed formulas in $PBL_r(3)$, but $B_i(1/2, \varphi)$ is not a well formed formula in $PBL_r(3)$.

The inner probabilistic model of $PBL_r(N)$ is also the same as PBL_r except that the measure assigned to every possible world should be the form of k/N respectively. Therefore, in an inner probabilistic model of $PBL_r(3)$, the measure in a possible world may be 1/3, 2/3 and etc, but can not be 1/2 or 1/4.

The inference system of $PBL_r(N)$ is also similar to PBL_r but all the probabilities in the axioms and inference rules should be the form of k/N respectively.

For example, *Axiom* 5 of PBL_r: "$B_i(a,\varphi) \to B_i(b,\varphi)$, where $1 \geq a \geq b \geq 0$" should be modified to "$B_i(a,\varphi) \to B_i(b,\varphi)$, where $1 \geq a \geq b \geq 0$ and a, b are the from of k_1/N, k_2/N" in *Axiom* 5 of the inference system of $PBL_r(N)$. Since the probabilities a and b in the formulas $B_i(a,\varphi)$ and $B_i(b,\varphi)$ are in the form of k_1/N, k_2/N, so the probability $max(a+b-1,0)$ in the scope of $B_i(max(a+b-1,0),\varphi \wedge \psi)$ in *Axiom* 2 and the probability $a+b$ in the scope of $B_i(a+b,\varphi \vee \psi)$ in *Rule* 4 are also in the form of k/N.

It is easy to see that the soundness of $PBL_r(N)$ holds.

Proposition 2. (Soundness of $PBL_r(N)$) If $\Gamma \vdash_{PBL_r(N)} \varphi$ then $\Gamma \models_{PBL_r(N)} \varphi$.

In the following, we prove the finite model property of $PBL_r(N)$. By this proposition, we can obtain the weak completeness of PBL_r immediately.

Definition 4. Suppose ζ is a consistent formula with respect to $PBL_r(N)$. $Sub^*(\zeta)$ is a set of formulas defined as follows: let $\zeta \in L^{PBL_r(N)}$, $Sub(\zeta)$ is the set of subformulas of ζ, then $Sub^*(\zeta) = Sub(\zeta) \cup \{\neg\psi | \psi \in Sub(\zeta)\}$. It is clear that $Sub^*(\zeta)$ is finite.

Definition 5. The model PM_ζ with respect to formula ζ is $(S_\zeta, P_{\zeta,1}, ..., P_{\zeta,n}, \pi_\zeta)$.

(1) Here $S_\zeta = \{\Gamma | \Gamma$ is a maximal consistent formulas set with respect to $PBL_r(N)$ and $\Gamma \subseteq Sub^*(\zeta)\}$.

(2) For any $\Gamma \in S_\zeta$, $P_{\zeta,i}(\Gamma) = (S_\zeta, X_\zeta, \mu_{\zeta,i,\Gamma})$, where $X(\varphi) = \{\Gamma' |\ \Gamma' \vdash_{PBL_r(N)} \varphi\}$ and $X_\zeta = \{X(\varphi) |\ \varphi$ is a Boolean combination of formulas in $Sub^*(\zeta)\}$; $\mu_{\zeta,i,\Gamma}$ is a mapping: $X_\zeta \to [0,1]$, and $\mu_{\zeta,i,\Gamma}(X(\varphi) = sup(\{a |$ $B_i(a,\varphi)$ is provable from Γ in $PBL_r(N)\})$.

(3) π_ζ is a truth assignment as follows: for any atomic formula p, $\pi_\zeta(p,\Gamma) = true \Leftrightarrow p \in \Gamma$.

The following lemmas show that the above model PM_ζ is an inner probabilistic model of $PBL_r(N)$, and it is canonical: for any $\Gamma \in S_\zeta$ and any $\varphi \in Sub^*(\zeta)$, $\varphi \in \Gamma \Leftrightarrow (PM_\zeta, \Gamma) \models \varphi$. This implies the finite model property of $PBL_r(N)$.

Lemma 1. S_ζ is a nonempty finite set.

Proof. Since the rules and axioms of $PBL_r(N)$ are consistent, S_ζ is nonempty. For $Sub^*(\zeta)$ is a finite set, by the definition of S_ζ, the cardinality of S_ζ is no more than the cardinality of $\wp(Sub^*(\zeta))$.

Lemma 2. X_ζ is the power set of S_ζ.

Proof. Firstly, since $Sub^*(\zeta)$ is finite, so if $\Gamma \in S_\zeta$ then Γ is finite. We can let φ_Γ be the conjunction of the formulas in Γ. Secondly, if $A \subseteq S_\zeta$, then $A = X(\vee_{\Gamma \in A} \varphi_\Gamma)$. By the above argument, we have that X_ζ is the power set of S_ζ.

Lemma 3 If φ is consistent (here φ is a Boolean combination of formulas in $Sub^*(\zeta)$), then there exists Γ such that φ can be proved from Γ, here Γ is a maximal consistent set with respect to $PBL_r(N)$ and $\Gamma \subseteq Sub^*(\zeta)$.

Proof. For φ is obtainable from the Boolean connective composition of formulas in $Sub^*(\zeta)$, therefore by regarding the formulas in $Sub^*(\zeta)$ as atomic formulas,

φ can be represented in disjunctive normal form. Since φ is consistent, there is a consistent disjunctive term in disjunctive normal form expression of φ, let such term be $\psi_1 \wedge ... \wedge \psi_n$, then φ can be derived from the maximal consistent set Γ that contains $\{\psi_1, ..., \psi_n\}$.

Lemma 4. For any $\Gamma \in S_\zeta$, $P_\zeta(\Gamma)$ is well defined.

Proof. It suffices to prove the following claim: if $X(\varphi) = X(\psi)$, then $\mu_{\zeta,i,\Gamma}(X(\varphi)) = \mu_{\zeta,i,\Gamma}(X(\psi))$. If $X(\varphi) = X(\psi)$, it is clear that $\vdash \varphi \leftrightarrow \psi$. For suppose not, $\varphi \wedge \neg\psi$ is consistent. By Lemma 3, there is Γ' such that $\varphi \wedge \neg\psi$ can be proved from Γ', therefore $\Gamma' \in X(\varphi)$ and $\Gamma' \notin X(\psi)$, it is a contradiction. Thus $\vdash \varphi \leftrightarrow \psi$. By rule: $\vdash \varphi \rightarrow \psi \Rightarrow \vdash B_i(a, \varphi) \rightarrow B_i(a, \psi)$, we get $\vdash B_i(a, \varphi) \leftrightarrow B_i(a, \psi)$, which means $\mu_{\zeta,i,\Gamma}(X(\varphi)) = \mu_{\zeta,i,\Gamma}(X(\psi))$.

Lemma 5. Let $Pro_{\zeta,i,\Gamma}(\varphi) = \{a | B_i(a, \varphi) \in \Gamma\}$, then $sup(Pro_{\zeta,i,\Gamma}(\varphi)) \in Pro_{\zeta,i,\Gamma}(\varphi)$.

Proof. By the construction of model, $Pro_{\zeta,i,\Gamma}(\varphi)$ is one of the numbers $0/N$, $1/N$,...,N/N. Since the set $0/N, 1/N, ..., N/N$ is finite, therefore $sup(Pro_{\zeta,i,\Gamma}(\varphi)) \in Pro_{\zeta,i,\Gamma}(\varphi)$.

Lemma 6. If $A \in X_\zeta$, then $0 \leq \mu_{\zeta,i,\Gamma}(A) \leq 1$. Furthermore, $\mu_{\zeta,i,\Gamma}(\emptyset) = 0$ and $\mu_{\zeta,i,\Gamma}(S_\zeta) = 1$.

Proof. By the construction of model, it is clear that $\mu_{\zeta,i,\Gamma}$ has the following property: if $A \in X_\zeta$, then $0 \leq \mu_{\zeta,i,\Gamma}(A) \leq 1$.

By rule: $\vdash \varphi \Rightarrow \vdash B_i(1, \varphi)$, it is clear $\mu_{\zeta,i,\Gamma}(S_\zeta) = 1$. By axiom: $B_i(0, \varphi)$, we get $B_i(0, false)$, so $\mu_{\zeta,i,\Gamma}(\emptyset) \geq 0$. By *Rule* 4 of PBL_r, we get $\mu_{\zeta,i,\Gamma}(S_\zeta) \geq \mu_{\zeta,i,\Gamma}(S_\zeta) + \mu_{\zeta,i,\Gamma}(\emptyset)$, so $1 \geq 1 + \mu_{\zeta,i,\Gamma}(\emptyset)$, which implies $\mu_{\zeta,i,\Gamma}(\emptyset) = 0$.

Lemma 7. If $A_1, A_2 \in X_\zeta$ and $A_1 \subseteq A_2$, then $\mu_{\zeta,i,\Gamma}(A_1) \leq \mu_{\zeta,i,\Gamma}(A_2)$.

Proof. Since $A_1, A_2 \in X_\zeta$ assume $A_1 = X(\varphi)$, $A_2 = X(\psi)$. If $X(\varphi) \subseteq X(\psi)$, by rule: $\vdash \varphi \rightarrow \psi \Rightarrow \vdash B_i(a, \varphi) \rightarrow B_i(a, \psi)$, we have $\mu_{\zeta,i,\Gamma}(A_1) \leq \mu_{\zeta,i,\Gamma}(A_2)$. Therefore if $A_1, A_2 \in X_\zeta$ and $A_1 \subseteq A_2$, then $\mu_{\zeta,i,\Gamma}(A_1) \leq \mu_{\zeta,i,\Gamma}(A_2)$.

Lemma 8. If $A_1, A_2 \in X_\zeta$ and $A_1 \cap A_2 = \emptyset$, then $\mu_{\zeta,i,\Gamma}(A_1 \cup A_2) \geq \mu_{\zeta,i,\Gamma}(A_1) + \mu_{\zeta,i,\Gamma}(A_2)$.

Proof. Since $A_1, A_2 \in X_\zeta$, assume $A_1 = X(\varphi)$, $A_2 = X(\psi)$. By rule: $\vdash \neg(\varphi \wedge \psi) \Rightarrow \vdash B_i(a_1, \varphi) \wedge B_i(a_2, \psi) \rightarrow B_i(a_1 + a_2, \varphi \vee \psi)$, where $a_1 + a_2 \leq 1$, we have $\mu_{\zeta,i,\Gamma}(X(\varphi) \cup X(\psi)) \geq \mu_{\zeta,i,\Gamma}(X(\varphi)) + \mu_{\zeta,i,\Gamma}(X(\psi))$. Therefore if $A_1, A_2 \in X_\zeta$ and $A_1 \cap A_2 = \emptyset$, then $\mu_{\zeta,i,\Gamma}(A_1 \cup A_2) \geq \mu_{\zeta,i,\Gamma}(A_1) + \mu_{\zeta,i,\Gamma}(A_2)$.

Lemma 9. For any $C, D \in X_\zeta$, $\mu_{\zeta,i,\Gamma}(C \cap D) \geq \mu_{\zeta,i,\Gamma}C) + \mu_{\zeta,i,\Gamma}(D) - 1$.

Proof. Since $C, D \in X_\zeta$, assume $C = X(\varphi)$, $D = X(\psi)$, by axiom: $B_i(a, \varphi) \wedge B_i(b, \psi) \rightarrow B_i(max(a+b-1, 0), \varphi \wedge \psi)$, we get $\mu_{\zeta,i,\Gamma}(X(\varphi) \cap X(\psi)) \geq \mu_{\zeta,i,\Gamma}(X(\varphi)) + \mu_{\zeta,i,\Gamma}(X(\psi)) - 1$.

Lemma 10. Let $B_i^-(\Gamma) = \{\Gamma' | \{\varphi | B_i(1,\varphi) \in \Gamma\} \subseteq \Gamma'\}$, then $\mu_{\zeta,i,\Gamma}(B_i^-(\Gamma)) = 1$.

Proof. For Γ is a finite formulas set, therefore $B_i^-(\Gamma) = X(\wedge_{B_i(1,\varphi_n)\in\Gamma}\varphi_n)$, by axiom: $B_i(a,\varphi) \wedge B_i(b,\psi) \rightarrow B_i(max(a+b-1,0), \varphi \wedge \psi)$, we have that $\wedge B_i(1,\varphi_n) \rightarrow B_i(1, \wedge\varphi_n)$, so $B_i(1, \wedge_{B_i(1,\varphi_n)\in\Gamma}\varphi_n)$ can be proved from Γ in $PBL_r(N)$, so $\mu_{\zeta,i,\Gamma}(B_i^-(\Gamma)) = 1$.

Lemma 11. Let $\Lambda_{i,\Gamma} = \{\Gamma' | P_{\zeta,i}(\Gamma) = P_{\zeta,i}(\Gamma')\}$, then $\mu_{\zeta,i,\Gamma}(\Lambda_{i,\Gamma}) = 1$.

Proof. Suppose $\Gamma' \in B_i^-(\Gamma)$. If $B_i(a,\varphi) \in \Gamma$, by rule: $B_i(a,\varphi) \rightarrow B_i(1, B_i(a,\varphi))$, we get $B_i(1, B_i(a,\varphi)) \in \Gamma$, for $\Gamma' \in B_i^-(\Gamma)$, hence $B_i(a,\varphi) \in \Gamma'$. If $\neg B_i(a,\varphi) \in \Gamma$, by rule: $\neg B_i(a,\varphi) \rightarrow B_i(1, \neg B_i(a,\varphi))$, we get $B_i(1, \neg B_i(a,\varphi)) \in \Gamma$, for $\Gamma' \in B_i^-(\Gamma)$, hence $\neg B_i(a,\varphi) \in \Gamma'$. Therefore $B_i(a,\varphi) \in \Gamma$ iff $B_i(a,\varphi) \in \Gamma'$, which means for any $A \in X_\zeta, \mu_{\zeta,i,\Gamma}(A) = \mu_{\zeta,i,\Gamma'}(A)$, so $\Gamma' \in \Lambda_{i,\Gamma}$, and furthermore $B_i^-(\Gamma) \subseteq \Lambda_{i,\Gamma}$. For $\mu_{\zeta,i,\Gamma}(B_i^-(\Gamma)) = 1$, we get $\mu_{\zeta,i,\Gamma}(\Lambda_{i,\Gamma}) = 1$ as desired.

Lemma 12. For any $\Gamma \in S_\zeta$, $P_{\zeta,i}(\Gamma)$ is a $PBL_r(N)$-inner probability space.

Proof. By Lemma 1 to Lemma 11, we can get the claim immediately.

Lemma 13. The inner probabilistic model PM_ζ is a finite model.

Proof. By the definition of S_ζ, the cardinality of S_ζ is no more than the cardinality of $\wp(Sub^*(\zeta))$, which means $|S_\zeta| \leq 2^{|Sub^*(\zeta)|}$.

Lemma 14. In the canonical $PBL_r(N)$-model PM_ζ, for any $\Gamma \in S_\zeta$ and any $\varphi \in Sub^*(\zeta)$, $\varphi \in \Gamma \Leftrightarrow (PM_\zeta, \Gamma) \models \varphi$.

Proof. We prove the lemma by induction on the structure of φ. In the following, we only prove that $B_i(a,\psi) \in \Gamma \Leftrightarrow (PM_\zeta, \Gamma) \models B_i(a,\psi)$.

If $B_i(a,\psi) \in \Gamma$, by the construction of PM_ζ, $\mu_{\zeta,i,\Gamma}(X(\psi)) = b \geq a$, we get $(PM_\zeta, \Gamma) \models B_i(a,\psi)$.

If $B_i(a,\psi) \notin \Gamma$, then $\neg B_i(a,\psi) \in \Gamma$, by the construction of PM_ζ and Lemma 5, since $sup(Pro_{\zeta,i,\Gamma}(\psi)) \in Pro_{\zeta,i,\Gamma}(\psi)$, so we have $sup(Pro_{\zeta,i,\Gamma}(\psi)) = b < a$, and $\mu_{\zeta,i,\Gamma}(X(\psi)) = b$, which implies $(PM_\zeta, \Gamma) \models \neg B_i(a,\psi)$, therefore $(PM_\zeta, \Gamma) \not\models B_i(a,\psi)$.

Proposition 3. (Finite model property of $PBL_r(N)$) If Γ is a finite set of consistent formulas, then there is a finite model PM such that $PM \models_{PBL_r(N)} \Gamma$.

Proof. By Lemma 14, there exists a finite $PBL_r(N)$-model $PM_{\wedge\Gamma}$ such that Γ is satisfied in $PM_{\wedge\Gamma}$.

Since any inner probabilistic model of $PBL_r(N)$ is also an inner probabilistic model of PBL_r, and any formula of PBL_r can be regarded as a formula of $PBL_r(N)$, given a consistent PBL_r-formula ζ, we can construct a $PBL_r(N)$-inner probabilistic model PM_ζ that satisfies formula ζ by the above lemmas. Since PM_ζ is also PBL_r-inner probabilistic model, so we can construct a PBL_r-inner probabilistic model PM_ζ that satisfies the given consistent PBL_r-formula ζ, this implies the finite model property of PBL_r.

Proposition 4. (Finite model property of PBL_r) If Γ is a finite set of consistent formulas, then there is a finite model PM such that $PM \models_{PBL_r} \Gamma$.

Proof. Let $a_1, a_2, ..., a_n$ be all rational numbers occur in the formulas in Γ. There are natural numbers $k_1, k_2, ..., k_n, N$ such that $a_i = k_i/N$. Firstly, since the axioms and rules of $PBL_r(N)$ is also the axioms and rules of PBL_r, therefore it is clear that if a finite set of formulas Γ is consistent with PBL_r, then it is also consistent with $PBL_r(N)$. By Proposition 3, there is a finite model of $PBL_r(N)$, PM, satisfying Γ. Since the model of $PBL_r(N)$ is also a model of PBL_r, so we get the proposition.

Proposition 5. (Weak completeness of PBL_r) If Γ is a finite set of formulas, φ is a formula, and $\Gamma \models_{PBL_r} \varphi$, then $\Gamma \vdash_{PBL_r} \varphi$.

Proof. Suppose not, then $(\wedge\Gamma) \wedge \neg\varphi$ is consistent with respect to PBL_r, by Proposition 4, there exists an inner probabilistic model $PM_{(\wedge\Gamma)\wedge\neg\varphi}$ such that $(\wedge\Gamma) \wedge \neg\varphi$ is satisfied in $PM_{(\wedge\Gamma)\wedge\neg\varphi}$, but this contradicts our assumption that $\Gamma \models_{PBL_r} \varphi$, thus the proposition holds.

From Proposition 4, we can get a procedure for checking if a formula φ is PBL_r-consistent. We simply construct every probabilistic model with $2^{|Sub^*(\varphi)|}$ states (Remember that in the construction of the finite model of φ, the values of inner probability measure are in the form of k/N, where N is a constant natural number. Since there are finite numbers having the form of k/N, where $0 \leq k \leq N$, therefore the number of inner probability measures assigned to the measurable sets is also finite, and consequently, the number of models with $2^{|Sub^*(\varphi)|}$ states is finite). We then check if φ is true at some state of one of these models. By Proposition 4, if a formula φ is PBL_r -consistent, then φ is satisfiable with respect to some models. Conversely, if φ is satisfiable with respect to some models, then φ is PBL_r-consistent.

As a consequence, we can now show that the provability problem for PBL_r is decidable.

Proposition 6. (Decidability of PBL_r) The provability problem for PBL_r is decidable.

Proof. Since φ is provable in PBL_r iff $\neg\varphi$ is not PBL_r-consistent, we can simply check if $\neg\varphi$ is PBL_r-consistent. By the above discussion, there is a checking procedure. Hence the provability problem for PBL_r is decidable.

3 Conclusions

In this paper, we proposed probabilistic belief logic PBL_r and gave the respective probabilistic semantics of this logic. We then presented an inference system of PBL_r. Furthermore, we proved the soundness property, the finite model property and the decidability of PBL_r. The above probabilistic belief logic allows the reasoning about uncertain information of agent in artificial intelligent systems.

References

1. Bacchus, F.: Representing and reasoning with probabilistic knowledge: a logical approach to probabilities. MIT Press, Cambridge, Mass (1990)
2. de C. Ferreira, N., Fisher, M., van der Hoek, W.: Practical Reasoning for Uncertain Agents. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 82–94. Springer, Heidelberg (2004)
3. de C. Ferreira, N., Fisher, M., van der Hoek, W.: Logical Implementation of Uncertain Agents. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 536–547. Springer, Heidelberg (2005)
4. Fagin, R., Halpern, J.Y.: Reasoning about knowledge and probability. J. ACM 41(2), 340–367 (1994)
5. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87(1/2), 78–128 (1990)
6. Halpern, J.Y., Tuttle, M.R.: Knowledge, probability, and adversaries. J.ACM 40(4), 917–962 (1993)
7. van der Hoek, W.: Some considerations on the logic PFD: A logic combining modality and probability. J. Applied Non-Classical Logics 7(3), 287–307 (1997)
8. Kooi, B.P.: Probabilistic Dynamic Epistemic Logic. Journal of Logic, Language and Information 12, 381–408 (2003)
9. Milch, B., Koller, D.: Probabilistic Models for Agent's Beliefs and Decisions. In: Proc. 16th Conference on Uncertainty in Artificial Intelligence 2000, pp. 389–396 (2000)

Towards Approximate BGI Systems

Barbara Dunin-Kęplicz[1] and Andrzej Szałas[2]

[1] Institute of Informatics, Warsaw University, Warsaw, Poland,
and ICS, Polish Academy of Sciences, Warsaw, Poland
keplicz@mimuw.edu.pl
[2] IDA, Linköping University, Linköping, Sweden
and WSIiE, Olsztyn, Poland
andsz@ida.liu.se

Abstract. This paper focuses on modelling perception and vague concepts in the context of multiagent BGI (*Beliefs, Goals* and *Intentions*) systems. The starting point is the multimodal formalization of such systems. Then we make a shift from Kripke structures to similarity structures, allowing us to model perception and vagueness in an uniform way, "compatible" with the multimodal approach. As a result we introduce and discuss *approximate* BGI *systems*, which can also be viewed as a way to implement multimodal specifications of BGI systems in the context of perception.

Keywords: multiagent systems, BGI systems, approximate reasoning, multimodal logics.

1 From Quantitative to Symbolic Modeling of MAS

The goal of any modelling of a reality is to create its adequate description. One of the most important factors, especially in the initial phases of modelling, is the choice of the underlying formalisms providing means for knowledge representation and reasoning adequately reflecting the environment and the application in question. In order to make a suitable choice, one should take into account the quality of available information, both when the model is being created and when it is used and acts on the basis on incoming data. There are many aspects that should be addressed. The current paper concentrates on questions concerning the quality of available information, including possible incompleteness, uncertainty and imprecision. The underlying formalism cannot adequately be chosen without considering such aspects.

Formal approaches to multiagent systems are concerned with equipping software agents with functionalities for reasoning and acting. The starting point of most of the existing approaches is the layer of beliefs. Usually, beliefs are represented in a symbolic, qualitative way. On the other hand, the basic layer of intelligent systems is the layer of perception. The perception results, like measurements from sensors, are inherently quantitative. Therefore we need to deal with a meta-level duality: sensors provide quantitative characteristics, while

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 277–287, 2007.

reasoning tasks require the use of symbolic representations and inference mechanisms. Thus, one of the challenges is to provide techniques that allow one to integrate perception with higher level knowledge structures which, in some cases, should be approximate.

BGI (*Beliefs, Goals* and *Intentions*) logics (see, e.g., [1,2,3,4,5,6]) may naturally serve as a high level specification of a multiagent system. It became clear, however, that commonly accepted basic methodological assumptions happen to be a subject of such substantial restrictions like:

- a high complexity of modal logics is well known and not easy to overcome
- the problem of incomplete and uncertain information has not been tackled
- the methods formalizing a commonsense reasoning are not incorporated.

In this paper we make a step towards introducing approximate reasoning into the multiagent context. More precisely, we propose the method of substituting multimodal approaches by a form of approximate reasoning suitable for modeling perception, namely a *similarity-based approximate reasoning*. We argue that this formalism allows one to both keep the intuitive semantics compatible with that of multimodal logics, and to model and implement phenomena occurring at the perception level.

1.1 Incompleteness Versus Imprecision

The lack of information creates a need to consider various variants of reality. In the course of acquiring information not feasible become excluded. In consequence, the total number of possible variants is usually shrinking.

This process does not naturally take place in the presence of uncertainty, where the knowledge might be complete, but still uncertain. For example, given that we do not know whether a specific object is a camel, once we find it out, we assume that our knowledge is certain and we do not consider other variants anymore. However, no matter how many measurements of its height we make using a short ruler, we can learn the result only up to a given precision which, at a certain point, cannot be improved. In other words, we cannot be sure what is the actual result of measurement: we can approximate it.

1.2 Approximate Versus Crisp

Traditional modelling is usually centered around crisp (precise) definitions of the modelled reality. The underlying assumption is that the objective reality is or can be crisply modelled. For example, the speed of an object is crisply defined as the rate of change of its displacement from a given starting point. Then one uses quantitative representation of distance and time and calculates the speed, assuming that the result is precise. Despite of the fact that numbers representing measurements might not be accurate, when one considers qualitative descriptions, like *fast* or *slow*, it becomes difficult to provide their precise definitions. We might classify some speeds to surely be fast, some to surely not be fast, while having difficulties to classify some others. According to the literature (see, e.g., [7]), a concept is *approximate* (vague) when it has borderline

cases, i.e., some objects cannot be classified to the concept or to its complement with certainty.

The environments in which intelligent systems are embedded are often much more complex and dynamic than the one already mentioned. Knowledge representation formalisms used to model such environments have to be partial and approximate in nature. Intelligent systems must be able to dynamically construct representations of observed objects and to integrate these representations with other static representations. Therefore McCarthy [8] argues that computer science logics should be able to deal with:

- approximate objects and concepts, which are not fully defined, lacking if-and-only-if definitions
- relations among approximate entities
- approximate theories which often involve approximate concepts and relations.

Example 1.1. In the BGI context, there are many situations, where one deals with vague concepts. For example, a team of robots can be given a goal to remove snow from a road and leave the road safe enough for driving. Apparently not only the concept "safe enough" is vague, also robots will most probably deal in this case with quite a lot of other approximate concepts as well as actions with approximate effects. Event the concept of "snow" becomes approximate, e.g., in conditions when it is in the middle of the melting process or, on the contrary, when it is freezing. ◁

As approximate concepts and relations are omnipresent in every physical world, any sensible description of the reality has to inherit this property. This is why they appear at practically all levels of intelligent systems dealing with real world applications:

1. on the object level
2. on the description (definition, specification) level
3. on the reasoning level.

The lowest (*object*) level deals with approximate objects of an application in question. Although approximativeness results from various reasons, in our analysis we will focus on the agent-level, having in mind that whatever complex techniques are recently developed, agent's activity starts from a perception. In our limited view on this process, we assume that perception captures the agent's ability to observe its environment. This observation functionality might be implemented in hardware, like video camera or an infrared sensor on a mobile robot, when agent is situated in the physical world.

Whatever form of perception is considered, it is hardly assumed to be perfect, as in most cases or most of the time its results are imprecise or, in the worst case, even misleading. Therefore it is increasingly important to fill gaps: between the access to low-level sensory data and its fusion as well as between the fused data and qualitative knowledge structures.

Next, the intermediate (*description*) level, consists of approximate predicates, approximate specifications (axioms) and approximate rules.

Finally, the *reasoning* level reflects reasoning involving approximate predicates and also approximates precise predicates when the reasoning process cannot fully be carried out, e.g., due to a high complexity of lack of sufficient resources.

1.3 A New Perspective

The starting point of our approach is the multimodal approach to modelling BGI systems ([1,2,3,5,6]). Such approaches allow one, among others, to specify and verify subtle properties concerning agents' cooperation, coordination, coalition formation, communication, etc. A typical property expressed within such frameworks could be

$$Bel_j[Int_i(keep\_low\_temperature)] \rightarrow Goal_j(keep\_low\_pressure), \qquad (1)$$

meaning that if the agent j believes that ith agent adopts a goal to keep the temperature low then jth agent's goal is to keep the pressure low.

In the current paper we make a shift in the perspective. Assuming perception and vague concepts are present, we provide a pragmatic machinery for agents to verify whether a particular state of the world, actual or hypothetical, *surely* or *possibly* satisfies their beliefs, goals and intentions. For example, we will be able to express properties like

$$low\_temperature_{\sigma_t{}^+} \wedge low\_pressure_{\sigma_p{}^\oplus}$$

meaning that, assuming that the accuracy of sensors modelled by similarities σ_t, σ_p, the temperature is surely low and the pressure might be low.

The immediate gain is that we deal with the first-order formalism which allows one to embed typical models of sensors[1] and perception, as well as to deal with vague concepts (for a discussion how to approach vague concepts within the similarity based framework, see also [10]). Observe that specification expressed as (1) contains vague concepts which have to be further modelled.[2] Assuming that agents actually work over relational or deductive databases and that similarities themselves are tractable, we stay within a very pragmatic, tractable framework, allowing us to implement multimodal BGI specifications in an efficient manner.

The paper is structured as follows. Section 2 discuses the BGI paradigm for modelling multiagency. Section 3 recalls preliminaries concerning similarity spaces. In Section 4 we introduce and discuss approximate BGI systems. Finally, Section 5 concludes the paper.

[1] For the idea of embedding sensors' models within the similarity spaces framework, see [9].

[2] Vagueness in the context of modal logics has been discussed , e.g., in [11], but this approach is not very natural for modelling perception.

2 Multimodal Models of BGI Systems

The BGI model of agency comprises beliefs referring to agent's informational attitudes, goals (or desires), intentions and then commitments referring to its motivational attitudes. A "mental state" of an BGI-agent is characterized by:

- *beliefs* - information the agent has about its environment
- *goals* - "options" available to the agent, i.e., different states of affairs that the agent may choose to commit to
- *intentions* - consistent states of affairs that the agent has chosen.

In [1,2,3] intentions initiate a goal-directed activity reflected in *commitments*.

The theory of informational attitudes has been formalized in terms of epistemic logic by [5,6]. As regards motivational attitudes, the situation is much more complex. In Distributed Cooperative Problem Solving, a group as a whole needs to act in a coherent pre-planned way, presenting a unified *collective* motivational attitude. This attitude, while staying in accordance with individual attitudes of group members, should have a higher priority than individual ones. Thus in BGI systems these attitudes are considered on three levels: individual, bilateral, and collective. A coherent and conceptually unified theory of motivational attitudes has been developed (see, e.g., [1,2,3]).

Traditionally, BGI systems have been formalized in terms of multi-modal logics tailored to model agents' informational and motivational attitudes (see, e.g., [1,2,3,4]). Beliefs, goals and intentions can naturally be modelled by the use of modal operators. Their semantics is based on Kripke structures and provides a general modelling machinery. However, already on an individual level of agent's specification there is a problem of adequate modeling of, e.g., beliefs and intentions, whenever limited perception capabilities, uncertain information and approximate information is present.

Example 2.1. Assume that I have a crisp belief that airplanes fly. If an object is perceived and can surely be classified as an airplane, the derived belief that it flies is also sure. If perceptual limitations do not allow to unambiguously classify the object as an airplane, but do not exclude it, the derived belief that it flies is uncertain. ◁

Therefore, despite many advantages, (multi)modal logics are often questioned in cases when *complexity* is an issue, when *perception* modelling is substantial and when *vague concepts* over (pseudo) continuous/dense domains are present. Especially, in multiagent environments we usually deal with agents' (robots') bounded resources and limited precision of sensors, video cameras and other equipment, approximative nature of real-world environments is either not adequately captured by modal theories or the constructed models become too complex.

Example 2.2. Consider a situation when we have two sensors:

- τ for measuring temperature with an associated predicate $T(t)$ ($T(t)$ is TRUE when the temperature is approximately t)

– π for measuring pressure with an associated predicate $P(p)$ ($P(p)$ is TRUE when the pressure is approximately p).

We know that measurement errors are not greater than ϵ_τ for τ and ϵ_π for π.

Let us assume that the situation *is safe* when $t < 60$ or when $60 \leq t \leq 100$ and $p \leq 4$. We denote this property by $safe(t, p)$. We then have:

$$safe(t, p) \stackrel{\text{def}}{\equiv} t < 60 \ \vee \ \big[60 \leq t \leq 100 \ \wedge \ p \leq 4\big]. \tag{2}$$

In this scenario:

– beliefs concern the accuracy of sensors and actions according to the specification given above
– there might be many goals, including keeping the situation safe (expressed by (2)), keeping the first disjunct ($t < 60$) of (2) satisfied, keeping the second disjunct ($60 \leq t \leq 100 \ \wedge \ p \leq 4$) of (2) satisfied, etc.
– we assume that our current intention[3] is to keep the temperature below 60^oC, i.e., that $t < 60$.

However, can we really use these crisp definitions when sensors' measurements can deviate from reality? For example, is the situation safe, when $t = 100$ and $p = 4$? When answering this question, measurement errors have to be taken into the consideration (see the continuation of this discussion provided in Example 4.1). ◁

The contribution of this paper is a successful replacement of modal logics by similarity spaces [12,13], keeping "compatible" and intuitive semantics, while *substantially simplifying calculations and reducing the complexity of reasoning.* The method we propose in the sequel is based on the following observations:

1. many agent theories are expressed in multimodal logics
2. there is a natural correspondence between modal theories and similarity structures
3. similarity spaces can be used to define approximations of concepts and relations
4. approximations of concepts and relations lead to approximate reasoning, tractable over relational and deductive databases (when one uses traditional tractable querying machinery, which is described in depth, e.g., in [14]).

3 Similarity Spaces

3.1 Preliminaries

There is a natural generalization of crisp relations by means of rough sets and relations, as introduced in [15]. These can further be generalized to approximate relations based on similarity spaces. In order to approximate these relations one uses a covering of the underlying domain by similarity-based neighborhoods (see, e.g., [12,13]). In this approach the lower and upper approximations of relations

[3] Recall that intentions are consistent subsets of goals.

are defined via neighborhoods rather than equivalence classes which are used in the case of rough sets. Approximate relations and similarity spaces have been shown to be quite versatile in many application areas [16,9,10].

There are many choices of possible constraints on the similarity relations used to define upper and lower approximations. For example, in some situations one might want transitivity, while in some others preferring similarities not to be transitive since similar objects might not naturally chain in a transitive manner. Many of these issues have been discussed in the context of rough sets (see, e.g., [15,17,18]). Therefore, for the sake of generality, we do not place any initial constraints on similarity relations.

Definition 3.1. *By a* similarity space *we mean any pair* $\langle U, \sigma \rangle$*, where* U *is a non-empty set and* $\sigma \subseteq U \times U$*. By a* neighborhood *of* u *wrt* σ *we mean* $n^\sigma(u) \stackrel{\text{def}}{=} \{u' \in U \mid \sigma(u, u')\}$*. For* $A \subseteq U$*, the* lower and upper approximation of A wrt σ*, denoted by* A_{σ^+} *and* $A_{\sigma^\oplus}$*, are defined by* $A_{\sigma^+} = \{u \in U\colon n^\sigma(u) \subseteq A\}$, $A_{\sigma^\oplus} = \{u \in U\colon n^\sigma(u) \cap A \neq \emptyset\}$.[4] ◁

Let $S = \langle U, \sigma \rangle$ be a similarity space and let $A \subseteq U$. Then

- A_{S^+} contains elements which *surely belong* to A (since all elements similar to the considered one belong to A)
- $A_{S^\oplus}$ contains elements which *may or may not belong* to A but it is impossible to determine that (due to the indiscernibility of similar objects) *in addition to elements which surely belong* to A.

Proposition 3.2. *Let* $S = \langle U, \sigma \rangle$ *be a similarity space. Then:*

$$A_{S^+} = \{u \in A \mid \forall v\, [\sigma(u, v) \rightarrow v \in A]\}$$
$$A_{S^\oplus} = \{u \in A \mid \exists v\, [\sigma(u, v) \wedge v \in A]\}.$$ ◁

Observe the strong similarity between characterizations of lower and upper approximations provided in Proposition 3.2 and definitions of semantics of $\Box$ and $\Diamond$ in Kripke semantics of modal logics.

3.2 Kripke Structures for Similarity-Based Reasoning

An obvious way to construct the accessibility relation R for similarity-based reasoning is to take the similarity relation σ and set

> $R(w, w') \stackrel{\text{def}}{\equiv}$ w differs from w' on the value of at most one variable (tuple of variables), say x, such that the value of x in w is similar to the value of x in w', i.e., when $\sigma(w(x), w'(x))$ holds, where $w(x)$ stands for the value of x in w.

Thus we do not really need Kripke structures, is underlying similarity spaces are known. Moreover, when similarities are defined over continuous domains, we rather want to avoid the continuum of accessible worlds. In fact, the similarity relation provides entire information about such Kripke structures. Therefore it suffices to specify one (current) world equipped with a suitable similarity spaces.

[4] One can also define other useful operators, like $A_{\sigma^-} \stackrel{\text{def}}{=} -A_{\sigma^\oplus}$ and $A_{\sigma^\pm} \stackrel{\text{def}}{=} -A_{\sigma^+} \cap -A_{\sigma^-}$.

Remark 3.3. Observe that one of the differences between multimodal approaches to BGI systems and the approach we propose is that the underlying accessibility relations model beliefs, goals and intentions separately, while similarity spaces do not have to be specific for beliefs, goals or intentions. A typical situation is that the same collection of similarity spaces, reflecting perception, serves for modelling beliefs, goals and intentions. However, other possibilities are not excluded. For example, the database with approximate beliefs could have been constructed in different circumstances (reflected by different similarity spaces) than those when specific goals are to be achieved. ◁

3.3 Correspondences Between Approximations and Similarity

As observed in Proposition 3.2, there is a close relationship between correspondences between approximations and similarities and those considered in modal correspondence theory, when modal $\Box$ is considered as A_{S^+} and modal $\Diamond$ is considered as $A_{S^\oplus}$.

Let S be a similarity space and σ be the similarity relation induced by S. Then

1. $A_{S^+} \subseteq A_{S^\oplus}$ is equivalent to the seriality of σ
2. $A_{S^+} \subseteq A$ is equivalent to the reflexivity of σ
3. $A \subseteq (A_{S^\oplus})_{S^+}$ is equivalent to the symmetry of σ
4. $A_{S^+} \subseteq (A_{S^+})_{S^+}$ is equivalent to the transitivity of σ.

In [13] a technique allowing one to automatize the process of finding suitable correspondences, analogous to those used in modal correspondence theory, has been considered.

4 Approximate BGI Systems

In the most general setting, by *approximate* BGI *systems* we understand the BGI systems, where beliefs, goals, intentions as well as specifications of axioms are expressed by means of approximate theories.[5]

In general, since beliefs, goals and intentions are characterized in multimodal logics as modalities of the universal flavor ("boxes"), we translate them into lower approximations w.r.t. suitable similarity spaces. This way we obtain a characterization of situations that *surely* satisfy given beliefs, goals and intentions. This is a very natural way to specify such properties as $safe$. On the other hand, in the case of dual properties one is often interested in characterization of *possible* situations. For example, when considering dangers, one is interested when the situation *might be* dangerous rather than when it *surely* is such. Therefore we additionally use upper approximations to express what *might be* believed, intended or what *might be* a goal.

[5] Observe that by using approximate databases one gets a tractable querying machinery, provided that the similarity relation and all the underlying operations (like the arithmetical ones), if used, are tractable.

In order to ensure the adequate properties of accessibility relations in multimodal BGI logics, one has to make sure that the underlying similarity relations satisfy relevant semantical properties. If only approximation operators are given, without similarities themselves, one can use the approach outlined in [13] together with the correspondences between approximations and similarities.

Example 4.1 (Example 2.2 continued). Recall that we have considered two sensors τ for measuring temperature and π for measuring pressure with the measurement errors not greater than ϵ_τ and ϵ_π (for τ and π, respectively). We then have two similarity spaces, S_τ reflecting the measurement errors of τ and S_π reflecting the measurement errors of π. More precisely, S_τ can be defined, e.g., to be $\langle T, \sigma_\tau \rangle$, where $T \stackrel{\text{def}}{=} [-20, 180]$ and $\sigma_\tau(t, t') \stackrel{\text{def}}{\equiv} |t - t'| \leq \epsilon_\tau$. Similarly S_π can be defined, e.g., to be $\langle P, \sigma_\tau \rangle$, where $P \stackrel{\text{def}}{=} [0, 10]$ and $\sigma_\pi(p, p') \stackrel{\text{def}}{\equiv} |p - p'| \leq \epsilon_\pi$. In such a case,

$$[-20, 60)_{S_\tau^+} = [-20 + \epsilon_\tau, 60 - \epsilon_\tau) \qquad [-20, 60)_{S_\tau^\oplus} = [-20, 60 + \epsilon_\tau) \quad (3)$$

$$[60, 100]_{S_\tau^+} = [60 + \epsilon_\tau, 100 - \epsilon_\tau] \qquad [60, 100]_{S_\tau^\oplus} = [60 - \epsilon_\tau, 100 + \epsilon_\tau] \quad (4)$$

$$[0, 4]_{S_\pi^+} = [\epsilon_\pi, 4 - \epsilon_\pi] \qquad [0, 4]_{S_\pi^\oplus} = [0,\ 4 + \epsilon_\pi]. \quad (5)$$

Let us now start with approximating the intention we have considered in Example 2.2 (to keep the temperature below $60^o C$). As calculated in (3), this intention is surely satisfied when t is in the lower approximation of the interval $[-20, 60)$,[6] i.e., when $t \in [-20+\epsilon_\tau, 60-\epsilon_\tau)$, which can be expressed as $-20+\epsilon_\tau \leq t < 60-\epsilon_\tau$.

The situation is not always that simple. For example, how can we deduce that the situation is (surely) safe? Using characterizations provided as (3) and (5), one could be tempted to define the lower approximation of $safe$ to be

$$-20 \leq t < 60 - \epsilon_\tau \ \vee \ \big[60 + \epsilon_\tau \leq t \leq 100 - \epsilon_\tau \ \wedge \ \epsilon_\pi \leq p \leq 4 - \epsilon_\pi\big]. \quad (6)$$

However, formula (6) does not properly reflect this approximation. Figure 1 shows the correct approximations. In particular, when the temperature is 60 and pressure is 2, formula (6) is not satisfied, while such a point is actually in the lower approximation of $safe$. ◁

In Example 4.1 we dealt with combining similarity relations and using suitable calculus on approximations. The problem of combining similarity relations and using them within the framework of modal logics has been addressed, e.g., in [19,20]. However, reasoning based on these approaches is quite complex. On the other hand, when one deals with similarity spaces and relational or deductive databases, the situation becomes tractable, assuming that the underlying similarity relations are tractable. In order to compute the set of objects x satisfying A_{σ^+} one just queries the database using the classical first-order formula $\forall y[\sigma(x, y) \rightarrow A(y)]$, where y refers to objects (e.g., rows in database tables).[7] Similarly, computing $A_{\sigma^\oplus}$ depends on supplying query $\exists y[\sigma(x, y) \wedge A(y)]$ to the database.[8]

[6] The constraint $-20 \leq t$ reflects the assumed domain of temperatures.

[7] This query computes all values of x satisfying $\forall y[\sigma(x, y) \rightarrow A(y)]$.

[8] Computing all values of x satisfying $\exists y[\sigma(x, y) \wedge A(y)]$.

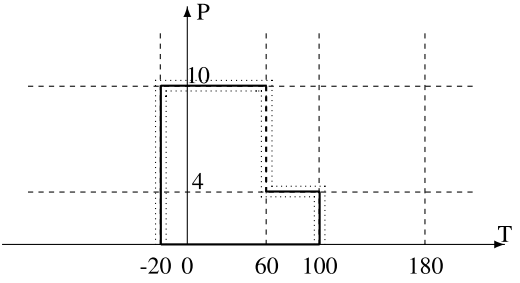


Fig. 1. The area representing the crisp definition of $safe$ (marked by the thick line) and its approximations (marked by dotted lines)

5 Conclusions

Many high-level knowledge structures are approximate per se, having both quantitative and qualitative characteristics. Agents have to deal with such structures. BGI systems are often formalized using multimodal logics. On the other hand, there is a natural way to model multimodal specifications within the similarity-based approximate reasoning framework, keeping the most important advantages of modal approach, while gaining lower complexity and intuitive construction of the underlying models, in particular in the presence of vague concepts and perception. A natural outcome of the presented research is a method for implementing multimodal specifications of BGI systems in a tractable, pragmatic manner.

It is also tempting to combine multimodal approaches with the approach based on similarity structures. In such a case one would deal with modalities reflecting sure and possible beliefs (as discussed in Example 2.1), sure and possible goals as well as sure and possible intentions, enhancing the existing approaches. There is an ongoing research on such a combination.

References

1. Dunin-Kęplicz, B., Verbrugge, R.: Collective intentions. Fundamenta Informaticae 51(3), 271–295 (2002)
2. Dunin-Kęplicz, B., Verbrugge, R.: Evolution of collective commitments during teamwork. Fundamenta Informaticae 56, 329–371 (2003)
3. Dunin-Kęplicz, B., Verbrugge, R.: A tuning machine for cooperative problem solving. Fundamenta Informaticae 63, 283–307 (2004)
4. Dziubiński, M., Verbrugge, R., Dunin-Kęplicz, B.: Complexity issues in multiagent logics. Fundamenta Informaticae 75(1-4), 239–262 (2007)
5. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge, MA (1995)
6. Meyer, J.J., van der Hoek, W.: Epistemic Logic for AI and Theoretical Computer Science. Cambridge University Press, Cambridge (1995)
7. Williamson, T.: Vagueness. Routledge, London (1994)

8. McCarthy, J.: Approximate objects and approximate theories. In: Cohn, A., Giunchiglia, F., Selman, B. (eds.) KR2000. Proc. 7th International Conf. on Principles of Knowledge Representation and Reasoning, pp. 519–526. Morgan Kaufmann Pub., Inc., San Francisco, Ca. (2000)
9. Doherty, P., Łukaszewicz, W., Szałas, A.: Communication between agents with heterogeneous perceptual capabilities. Journal of Information Fusion 8, 56–69 (2007)
10. Doherty, P., Łukaszewicz, W., Szałas, A.: Similarity, approximations and vagueness. In: Ślęzak, D., Wang, G., Szczuka, M., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3641, pp. 541–550. Springer, Heidelberg (2005)
11. Halpern, J.: Intransitivity and vagueness. In: Dubois, D., Welty, C., Williams, M.A. (eds.) Proc. of 9th Int. Conf. KR'2004, pp. 121–129. AAAI Press, Stanford, California, USA (2004)
12. Doherty, P., Łukaszewicz, W., Szałas, A.: Tolerance spaces and approximative representational structures. In: Günter, A., Kruse, R., Neumann, B. (eds.) KI 2003. LNCS (LNAI), vol. 2821, pp. 475–489. Springer, Heidelberg (2003)
13. Doherty, P., Szałas, A.: On the correspondence between approximations and similarity. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS (LNAI), vol. 3066, pp. 143–152. Springer, Heidelberg (2004)
14. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley Pub. Co., Reading (1996)
15. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
16. Doherty, P., Łukaszewicz, W., Skowron, A., Szałas, A.: Knowledge Representation Techniques. A Rough Set Approach. Studies in Fuziness and Soft Computing, vol. 202. Springer, Heidelberg (2006)
17. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
18. Słowiński, R., Vanderpooten, D.: A generalized definition of rough approximations based on similarity. IEEE Trans. on Data and Knowledge Engineering 12(2), 331–336 (2000)
19. Demri, S., Orłowska, E.: Incomplete Information: Structure, Inference, Complexity. Springer, Heidelberg (2002)
20. Doherty, P., Dunin-Kęplicz, B., Szałas, A.: Dynamics of approximate information fusion. In: Kryszkiewicz, M., Peters, J., Rybinski, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 668–677. Springer, Heidelberg (2007)

Verifying Dominant Strategy Equilibria in Auctions

Emmanuel M. Tadjouddine and Frank Guerin

Department of Computing Science, King's College,
University of Aberdeen, Aberdeen AB24 3UE, Scotland
{etadjoud,fguerin}@csd.abdn.ac.uk

Abstract. Future agent mediated eCommerce will involve open systems of agents interoperating between different institutions, where different auction protocols may be in use. We argue that in order to achieve this agents will need a method to automatically verify the properties of a previously unseen auction protocol; for example, they may wish to verify that it is fair and robust to deception. We are therefore interested in the problem of automatically verifying the game-theoretic properties of a given auction mechanism, especially the property of strategyproofness. In this paper we show how the `Alloy` model checker can be used to automatically verify such properties. We illustrate the approach via two examples: a simple two player Vickrey auction and a quantity restricted multi-unit auction using the Vickrey-Clarke-Groves mechanism.

1 Introduction

Future agent mediated eCommerce will involve open systems of agents interoperating between different institutions, where different auction protocols may be in use; the roaming agents will need to understand the rules of engagement in foreign institutions. This *semantic interoperation* for agents in different institutions is recognised as a "major challenge facing computer scientists" [1]. We look at the special case of agents engaged in auctions. Agent auctions have two aspects which we need to consider: (i) they specify a set of rules which the agents must follow, specifying when they are allowed to bid, how the winner should be determined and the payments to be made; (ii) they usually have certain desirable game theoretic properties such as incentive compatibility, encouraging agents to bid truthfully. Other properties can be *collusion-proofness* meaning agents cannot collude to achieve a certain outcome or *false-name bidding free* meaning agents cannot manipulate the outcome by using fictitious names. However, in here, we focus on the properties of a solution concept of the game, in particular dominant strategy equilibrium.

To enable agents to understand the rules of an unseen auction we need to consider both of these aspects. The first can be handled by agreeing on a standard language in which the rules of the auction can be written and published. There are a number of candidate languages in the literature, see for example [2]. The second aspect is considerably more problematic. Understanding the rules of

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 288–297, 2007.

an auction is not enough; an agent needs to know some of the properties of the auction (e.g. individual rationality, incentive compatibility) in order to make a decision about whether or not to participate, and what strategy to use. There is also a recursive nature to this understanding of the game-theoretic properties; most game theoretic solutions rely on the equilibrium property of a game, and this equilibrium rests on common knowledge assumptions; if the common knowledge of the equilibrium is not achieved, then agents cannot expect it to be played [3].

Assuming that the auction mechanism is published along with a claim that it is strategyproof (i.e. truthful bidding is optimal), we aim to verify that truthful bidding is indeed strategyproof. Such verification is important for selfish and rational agents engaged in financial transactions. An agent cannot just trust the auctioneer and then choose the recommended strategy. To illustrate the verification procedure, we consider the Vickrey auction and a quantity-restricted multi-unit auction (QRMUA) and verify these equilibrium properties via the `Alloy` model checker based on first-order relational logic, see `http://alloy.mit.edu/`.

The contributions of this paper are the following. i) We have provided a polynomial time algorithm solving the QRMUA, integrated it into the VCG mechanism, and proved some properties on the revenue of the resulting pricing rule. ii) We have also shown by example that there are certain restricted cases of combinatorial auctions where it is feasible for agents to automatically check the relevant game theoretic properties, and hence that it is feasible for agents to interoperate between different auction houses and work with previously unseen specifications, for simple auctions.

In Section 2, we introduce our notations, and describe the combinatorial auction problem and the Vickrey-Clarke-Groves (VCG) mechanism. In Section 3, we describe how model checking can be used to verify the game theoretic properties of an auction. In Section 4, we illustrate this approach using a simple Vickrey auction. In Section 5, we look at the case of a simple type of combinatorial auction; we formulate a winner determination algorithm for it, and check the game theoretic properties of a VCG implementation of it. Section 6 concludes.

2 Background and Model

In a *combinatorial auction*, there is a set I of m items to be sold to n potential buyers. A bid is formulated as a pair $(B(x), b(x))$ in which $B(x) \subseteq I$ is a bundle of items and $b(x) \in \mathbb{R}^+$ is the price offer for the items in x. Given a set of k bids, $X = \{(B(x_1), b(x_1)), (B(x_2), b(x_2)), \ldots, (B(x_k), b(x_k))\}$ the *combinatorial auction problem*(CAP) is to find a set $X_0 \subseteq X$ such that $\sum_{x \in X_0} b(x)$ is maximal, subject to the constraints that for all $x_i, x_j \in X_0 :\ B(x_i) \cap B(x_j) = \emptyset$ meaning an item can be found in only one accepted bid. We assume *free disposal* meaning that items may remain unallocated at the end of the auction.

Unfortunately the CAP is NP-hard [4]. Consequently, approximation algorithms [5] are used to find a near-optimal solution or restrictions are used to find tractable instances of the CAP [6] hence providing polynomial time

algorithms for a restricted class of combinatorial auctions. A combinatorial auction can be *sub-additive* (for all bundles $B_i, B_j \subseteq I$ such that $B_i \cap B_j = \emptyset$, the price offer for $B_i \cup B_j$ is less than or equals to the sum of the price offers for B_i and B_j) or *super-additive* (for all $B_i, B_j \subseteq I$ such that $B_i \cap B_j = \emptyset$, the price offer for $B_i \cup B_j$ is greater than or equals to the sum of the price offers for B_i and B_j).

To setup the auction, we use a game theory *mechanism* [7], a decision procedure that determines the set of winners for the auction according to some desired objective. An objective may be that the mechanism should maximise the social welfare. Such a mechanism is termed *efficient*. For open multi-agent systems, other desirable properties for the mechanism designer can be *incentive compatibility* (no bidder can benefit from lying provided all other agents are truthful) or *strategyproofness* (truthtelling is a dominant strategy). A mechanism that is strategyproof has a dominant strategy equilibrium. A well known class of mechanisms that is efficient and strategyproof is the Vickrey-Clarke-Groves (VCG), see for example [7,8]. The VCG mechanism is performed by finding (i) the allocation that maximises the social welfare and (ii) a pricing rule allowing each winner to benefit from a discount according to his contribution to the overall value for the auction. To formalise the VCG mechanism, let us introduce the following notations:

- $\mathcal{X}$ is the set possible allocations
- $v_i(x)$ is the true valuation of $x \in \mathcal{X}$ for bidder i
- $b_i(x)$ is the bidding value of $x \in \mathcal{X}$ for bidder i
- $x^* \in \text{argmax}_{x \in \mathcal{X}} \sum_{i=1}^{n} b_i(x)$ is the optimal allocation for the submitted bids.
- $x^*_{-i} \in \text{argmax}_{x \in \mathcal{X}} \sum_{j \neq i}^{n} b_j(x)$ is the optimal allocation if i were not to bid.
- u_i is the utility function for bidder i.

The VCG payment p_i for bidder i is defined as

$$\begin{aligned} p_i &= b_i(x^*) - \left(\textstyle\sum_{j=1}^{n} b_j(x^*) - \sum_{j=1,\ j \neq i}^{n} b_j(x^*_{-i}) \right) \\ &= \textstyle\sum_{j=1,\ j \neq i}^{n} b_j(x^*_{-i}) - \sum_{j=1,\ j \neq i}^{n} b_j(x^*), \end{aligned} \tag{1}$$

and then, the utility u_i for agent i is a quasi-linear function of its valuation v_i for the received bundle and payment p_i.

$$u_i(v_i, p_i) = v_i - p_i.$$

Let p_i^* and p_i be the payments for agent i when it bids its true valuation v_i and any number b_i respectively. Note p_i, p_i^* are functions of b_{-i}. The strategyproofness of the mechanism amounts to the following verification:

$$\forall i,\ \forall v_i,\ \forall b_i\ u_i(v_i, p_i^*(b_{-i})) \geq u_i(v_i, p_i(b_{-i})). \tag{2}$$

In the equation (1), the quantity $\sum_{j=1,\ j \neq i}^{n} b_j(x^*_{-i})$ is called the *Clark tax*. Another interesting property of this VCG mechanism is that it is *weakly budget balanced* [9] meaning the sum of all payments is greater than or equal to zero. For the VCG properties (e.g. strategyproof) to hold, the auctioneer must solve

n+1 hard combinatorial optimisation problems (the optimal allocation in the presence of all bidders followed by n optimal allocations with each bidder removed) exactly. In this work, we consider tractable instances of the CAP to which we apply the VCG mechanism. We formally specify the auction using a programming language, and then we verify some game-theoretic properties of the auction. These properties are simply first-order logic formulae within the auction model, e.g. the winner is always the highest bidder or the auction mechanism is incentive compatible or strategyproof.

3 Verifying Properties as Model Checking

To verify game-theoretic properties of a given auction, we use the `Alloy` model checker based on first order relational logic [10,11]. We then model the basic entities of the auction as well as the relations between them. Next, we express the operations and the auction properties that we wish to check. To verify a property, the `Alloy` analyser starts by negating it and then by looking for an instance where the negated property is true within a range of small number of models according to a user-defined scope. If the model is a correct representation of the auction, such an instance represents a counter-example to the auction property. If no counter-example is found, the negated formula may still be true in a larger scope since first-order logic is undecidable and the `Alloy` tool achieves tractability by restriction to a finite universe. It is worth noting the `Alloy` analysis is intractable asymptotically but according to Daniel Jackson's *small scope hypothesis* "Many bugs have small counter-examples". In other words, "negative answers tend to occur in small models already, boosting the confidence we may have in a positive answer" [12, p. 143].

4 A Motivating Example: The Vickrey Auction

In this section we motivate the problem by encoding the Vickrey auction [13] using the `Alloy` modelling language. We show how an agent can employ simple IF-constructs to negate the well known game theoretic properties of the Vickrey auction, e.g. incentive compatibility; hence an agent would be able to check certain auction properties before entering it.

4.1 Modelling the Vickrey Auction

In a Vickrey auction, n agents bid for a single item. Each agent has a private valuation v of the item. The winner is the highest bidder; she gets the item but pays the second highest bid p, getting the utility $u{=}v{-}p$. A losing bidder pays nothing and has a zero utility. It is well known that this mechanism is strategyproof. We wish to enable the potential bidders to check such a claim for instance.

For clarity, we omit the details of the `Alloy` coding but provide the key stages of the modelling. Assuming bids and valuations are non negative integers, the Vickrey auction for two bidders is modelled in `Alloy` using a *signature* (type

structure or a class in the object oriented jargon) `Bidder`. A bidder is an entity p of type `Bidder` having a single valuation $\mathrm{val}(p)$, a bid $\mathrm{bid}(p)$, a boolean variable $w(p)$ indicating if it wins or loses, and a utility $u(p)$ associated with the outcomes of the auction. The model is constrained so as bids and valuations of the bidder are within a finite interval. Once the basic entities of our system are modelled, the winner determination algorithm and the utility function are coded as an `Alloy` predicate $\mathrm{wda}(p_1, p_2, p_1', p_2')$. This takes as arguments two bidders p_1, p_2 before the run of the auction and two bidders p_1', p_2' after. The bids and values of the two bidders remained unchanged. However, their utilities are computed according to the Vickrey mechanism using an IF-construct with a test on who has the highest bid.

4.2 Checking Auction Properties

Having modelled the auction and its operations describing its dynamic behaviour, we can ask for example the following questions. Is the winner always the highest bidder? Is the mechanism strategyproof? Such properties can be checked using `Alloy` *assertions*, which are first-order logic formulae; the `Alloy` analyser can check whether they are valid by seeking their counter-examples. The assertion `winIsHighBidder` can be expressed as follows:

$$\forall p_1, p_1', p_2, p_2' \texttt{:Bidder}, \mathrm{wda}(p_1, p_2, p_1', p_2') \wedge w(p_1') = 1 \rightarrow \mathrm{bid}(p_1') \geq \mathrm{bid}(p_2')$$

meaning after the run of the auction, the winner has the highest bid. The assertion `isDSE` for checking the strategyproofness can be expressed as follows:

$$\begin{array}{l}\forall p_1, p_2, p_2^t, p_1^* \texttt{:Bidder}, \forall p_1', p_2', p_2^{t'}, p_1^{*'} \texttt{:Bidder}, \\ (\mathrm{bid}(p_1^*) = \mathrm{val}(p_1^*) \wedge \mathrm{val}(p_1^*) = \mathrm{val}(p_1) \wedge \mathrm{val}(p_2^t) = \mathrm{val}(p_2) \wedge \mathrm{bid}(p_2^t) = \mathrm{bid}(p_2) \\ \wedge \mathrm{wda}(p_1, p_2, p_1', p_2') \wedge \mathrm{wda}(p_1^*, p_2^t, p_1^{*'}, p_2^{t'})) \rightarrow u(p_1^{*'}) \geq u(p_1').\end{array}$$

This considers 4 bidders p_1, p_2, p_1^*, p_2^t where p_2^t is a clone of p_2 and p_1, p_1^* have the same valuation but p_1^* bids truthfully. It then checks the utilities of bidders p_1, p_1^* after the run of the auction, say $p_1', p_1^{*'}$. Eventually, we used `Alloy` *check* commands to verify each assertion within a finite scope. For example, the `isDSE` assertion is checked for 50 values and 4 `Bidder` types.

This analysis was carried out on a PC Pentium Dual Processor 2.99 GHz with 2GB RAM, running Windows XP. The two assertions were found to be valid within the specified scope, which makes us believe their correctness. To give an idea of the runtime, the `isDSE` analysis took 312 CPU seconds to complete. But how much data have been processed? With 50 integers, there are 50^4 possible values for each bidder. An exhaustive search for all 4 bidders give 50^{16} cases to consider. Fortunately, `Alloy` uses tree optimisation techniques that can discard large search subspaces without fully inspecting them [10,11].

Interestingly, if we use the conditional $(\mathrm{bid}(p_1) \geq \mathrm{bid}(p_2) \vee \mathrm{bid}(p_2){=}2)$ to determine the highest bidder in the winner determination algorithm `wda`, then neither of the two assertions above is valid. For example, `Alloy` took 0.2 CPU second to find the counter-example shown in Figure 1, for the model checking

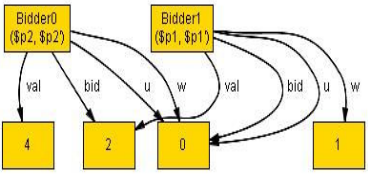


Fig. 1. Counter-example of the assertion `winIsHighBidder`

of `winIsHighBidder`. Figure 1 shows two bidders `Bidder0` valuing the item for 4 and bidding 2 and `Bidder1` valuing the item for 2 and bidding 0. `Bidder1` is the highest bidder but loses the auction as indicated by its field w=0.

5 Quantity-Restriction in Multi-unit Auctions

Quantity-Restriction in Multi-Unit Auctions (QRMUA) is a combinatorial auction in which each potential buyer submits bids for a number of identical items but restricts the number of items he wishes to buy. It has the property that the price offer for a bundle by a buyer is the sum of her bids on the individual items in the bundle. It is also sub-additive since a buyer will get for free any extra item allocated to him beyond the number of items she wished to obtain. Obviously, there is a maximum number of items to be allocated by the auctioneer. This is a special case of the quantity-restriction in multi-object auctions of [6], which is shown to be tractable by reducing it to a b-matching problem [14, p. 186] in a bipartite graph. The QRMUA can be illustrated by the auction example for reservation of seats in a particular flight given in [6].

Let m be the size of I e.g., the set of seats in the airplane and n the number of potential buyers with their required numbers of seats $m_1, m_2, \ldots, m_n$ respectively. The price offer for each bundle of m_i seats is $m_i * b_i$, wherein b_i is the price offer for a single seat. Let a_i be the number of seats allocated to agent i. The allocation consists in choosing disjoint subsets $\{B_i \subseteq I | i \in \mathit{Winners}\}$ of winning bids such that $\sum_{i \in \mathit{Winners}} a_i b_i$ is maximal, where $a_i = \min(|B_i|, m_i)$. This allocation can be formulated as the following Integer Linear Programming (ILP) problem:

$$\max \sum_{i=1}^{n} a_i b_i \quad \text{subject to} \begin{cases} a_i \leq m_i, \ i = 1, n \\ \sum_{i=1}^{n} a_i \leq m \end{cases} \tag{3}$$

Because ILP is generally intractable, we may use a similar argument to that of [15] to relax the ILP to linear programming, which is tractable, and then show that the resulting solution is integral. We recover the tractability result by providing a simple polynomial algorithm (see Algorithm 1) that solves the optimisation problem of equation (3).

Our algorithm takes as inputs n bids $b_1, b_2, \ldots, b_n$, n values $m_1, m_2, \ldots, m_n$ representing the numbers of required seats for each bidder and the total number

of seats m. It returns an allocation $a_1, a_2, \ldots, a_t$ that maximises the quantity $\sum_{i=1}^{n} a_i b_i$. This enables us to easily encode QRMUA into `Alloy`.

Algorithm 1 .(Allocation for QRMUA)
Sort $b_i, i = 1, n$ in decreasing order
Assume $b_1 \geq b_2 \geq \ldots \geq b_n$
Set $i := 1$; $r := m$; $s := 0$
While ($r > 0$ and $i \leq n$) Do
    If $r \geq m_i$ Then
        Set $a_i := m_i$; $r = r - a_i$
    Else
        Set $a_i := r$; $r := 0$
    End If
    Set $s := s + a_i b_i$; $i = i + 1$
End Do
Set $t := i - 1$
If $r > 0$ Then
    Choose randomly j, $1 \leq j \leq t$
    Set $a_j := a_j + r$
End If
At the end, return the values $(a_i)_{i=1,t}$ and the sum s

Lemma 1. *Algorithm 1 solves the ILP of equation (3) in $\mathcal{O}(n \log n)$ time.*

Proof: Algorithm 1 can sort the n bids in $\mathcal{O}(n \log n)$ using the quicksort procedure and then takes at most n steps to terminate. Thus, its complexity time. We need to prove this algorithm correctly solves optimisation problem of equation (3). The proof is by induction on the number m of items. The case m=1 is trivial. Assume the algorithm is correct for m items, we shall prove it remains so for m+1 items. Our assumption means we have found an allocation $a_1, a_2, \ldots, a_t$ such that $s = \sum_{i=1,t} a_i b_i$ is maximum with $b_1 \geq b_2 \geq \ldots \geq b_n$. Let us consider the case $t < n$. We have $a_i = m_i$ for $i = 1, \ldots, t-1$. If $a_t < m_t$ then our algorithm will find the value of a_t as being $a_t + 1$ and the sum s becomes $s + b_t$ with b_t the maximum value of the remaining bids $b_t, \ldots, b_n$. Therefore, $s+b_t$ is the maximum possible we can get w.r.t. the constraints of the problem (any seat allocated to a bidder beyond his required number of seats is free for that bidder). If $a_t = m_t$ then, the m+1th seat will be allocated to the next highest bidder t+1 and the sum becomes $s+b_{t+1}$, which is the maximum possible. Similarly, we can show that the case $t = n$ leads to the optimal allocation with the sum $s+b_n$ if $a_n < m_n$ or simply the sum s otherwise. □

5.1 The VCG Mechanism Applied to QRMUA

Assuming each bidder has a private valuation v_i and bids b_i for a single seat, we can use Algorithm 1 to find the n+1 optima x^* and $x^*_{-i}, i = 1, \ldots, n$ in the VCG payments of equation (1). To illustrate, let us consider the case where

$n = 2, m = 8, m_1 = 5, m_2 = 4$. Assume bidders bid their true valuations $b_1{=}v_1{=}4$ and $b_2{=}v_2{=}3$. We have the following payments:

$$p_1 = b_2(x^*_{-1}) - b_2(x^*) = 4 * 3 - 3 * 3 = 3$$
$$p_2 = b_1(x^*_{-2}) - b_1(x^*) = 5 * 4 - 5 * 4 = 0$$

Bidder 2 gets its bundle for free and the auctioneer makes the modest revenue of 3.

Lemma 2. *By applying the VCG mechanism to QRMUA, we have:*

1. *If $m_1 + m_2 + \ldots + m_n \leq m$ then the VCG payments are zero. The bidders get the bundles for free.*
2. *If $m_1 + m_2 + \ldots + m_n > m$ then there exists a VCG payment that is greater than zero.*

Proof: For the first part of the lemma, the total sum from the optimal allocation in the presence of all bidders is $\sum_{j=1}^{n} m_j b_j$. The total sum from the optimal allocation if i is not bidding is $\sum_{j\neq i}^{n} m_j b_j$ and the reported bid for i is $m_i b_i$. Consequently the VCG payments $p_i = 0$. For the second part of the lemma, let us assume without loss of generality $b_1 \geq b_2 \geq \ldots \geq b_n$, we shall prove that $p_1 > 0$. Since $m_1 + m_2 + \ldots + m_n > m$, Algorithm 1 will find an optimal allocation $x^* = a_1, \ldots, a_t$ with $t \leq n$ such that $\sum_{i=1}^{t} a_i = m$ and an optimal allocation $x^*_{-1} = a'_2, \ldots, a'_{t'}$ with $t \leq t' \leq n$ such that $\sum_{i=2}^{t'} a'_i = m$. We then have the payment $p_1 = \sum_{j=2}^{t'} a'_j b_j - \sum_{j=2}^{t} a_j b_j$. Let us consider the case $t' = t$. It follows $a_j = a'_j = m_j$ for $j = 2 \ldots t{-}1$ and $a'_t = a_1 + a_t$. Consequently $a'_t > a_t$ hence $p_1 = (a'_t - a_t)b_t > 0$. If $t' > t$, then we have $a_j = a'_j = m_j$ for $j = 2 \ldots t$. Consequently $p_1 = \sum_{j=t+1}^{t'} a'_j b_j > 0$. □

This low or even zero revenue for the auctioneer is one of the shortcomings of the VCG mechanism, see [16] for more VCG criticisms. Ideally, an auctioneer aims to get maximum revenue. Nonetheless, this VCG mechanism with the Clark tax has the attractive property of having a weak budget balance, which is vital in an auction setting. For an optimal allocation $a_1, a_2, \ldots, a_t$ with $t \leq n$, the winning bidders get the utilities $u_i = a_i v_i - p_i$. The losing bidders have zero utility. Having setup a tractable CAP, typically the QRMUA with an allocation and pricing rules, we wish to prove certain properties related to the underlying mechanism by using the `Alloy` modelling language.

5.2 Analysis of the Resulting Auction

As seen for the Vickrey auction, we model the basic entities, their relationships and the dynamic behaviour (the winner determination algorithm) of the resulting QRMUA within `Alloy` as follows. The items are identical and are represented by an `Alloy` signature `Item`. An `Alloy` model of the bidder contains a field `bid`, which is a set of pairs of bundle and price, along with a private value `val` for each individual item, a number of allocated items `alloc` and a utility `u`, which is

an integer. Note that, for this particular auction, the bids are additive since the price of a bundle is simply the sum of the prices of its individual items. Furthermore, the complexity of expressing additive bids in the OR bidding language is linear, see for example [15]. Therefore, we can use the OR language to express each agent's preferences. The winner determination algorithm is implemented by a predicate wda that determines the allocation for each bidder, its VCG payment and its associated utility. We then wish to verify the implemented auction mechanism is strategyproof or has a dominant strategy equilibrium.

5.3 Verifying Dominant Strategy Equilibrium

As shown for the Vickrey auction, the dominant strategy equilibrium property for QRMUA is expressed by a first order logic formula. Unlike the Vickrey auction, here we add two conditionals expressing the fact that the two sets of items allocated to the two bidders form a partition of the set of all items.

The strategyproofness property was verified for different scopes. To give an idea of the runtime, its verification for an auction composed of 20 items took 7 min and 10 s of CPU time to complete. Though, this is not a complete proof of correctness, this suggests that the property is likely to hold for an unbounded model. More importantly, whenever we introduce flaws in the auction mechanism, `Alloy` found a counter-example. This kind of checking is exponential in the number of players as shown by the following proposition.

Proposition 1. *Given a normal form game with n agents having d possible bidding values each and a recommended strategy profile $\mathbf{s}^*$, verifying that $\mathbf{s}^*$ is a dominant strategy equilibrium is in $\mathcal{O}(nd^n)$.*

Proof sketch: In the worst case scenario, each player's utility is affected by the $n-1$ remaining players. Scanning all different strategy profiles in order to check strategyproofness leads to a time complexity $\mathcal{O}(nd^n)$, which is exponential in the number of players. □

It is worth noting that by expressing the auction mechanism and its game-theoretic properties using first order logic, the checking remains undecidable in general. Naturally, we can restrict to a subset of first order logic that is decidable, e.g., the Presburger formulae [17]. However, the `Alloy` approach combining model checking that is tractable, with first order logic that is undecidable, enables us to find counter-examples for wrong formulae and to prove likely correct properties for some finite scope.

6 Conclusions

In this paper we have explained how an agent in an eCommerce scenario could automatically check some interesting properties of a published auction specification. The main property of interest is strategyproofness (i.e. the protocol is immune to counterspeculation and strategic bidding). Our approach to verification uses the Alloy model checker. In particular we have shown, by example, that

there are certain restricted cases of combinatorial auctions where it is feasible for agents to check for strategyproofness. The main lesson learnt is that while verifying this property is feasible, it is quite computationally difficult even for relatively simple auction types. In the future we intend to undertake a theoretical investigation of the limits of this type of automated property checking. We expect that some classes of auctions will prove difficult to verify automatically, and that some auctions will only be verifiable if their complexity is limited (e.g. limited number of items, bidders and bundle types). When these limits are determined, they can be used to inform the design of mechanisms for agent scenarios where semantic interoperation is desired.

References

1. Dash, R.K., Jennings, N.R., Parkes, D.C.: Computational-mechanism design: A call to arms. IEEE Intelligent Systems, 40–47 (November 2003)
2. Koller, D., Pfeffer, A.: Representations and solutions for game-theoretic problems. Artificial Intelligence 94(1), 167–215 (1997)
3. Guerin, F., Tadjouddine, E.M.: Realising common knowledge assumptions in agent auctions. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Hong Kong, China, pp. 579–586 (December 2006)
4. Rothkopf, M.H., Pekec, A., Harstad, R.M.: Computationally manageable combinatorial auctions. Management Science 44(8), 1131–1147 (1998)
5. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Winner determination in combinatorial auction generalizations. In: AAMAS, pp. 69–76 (2002)
6. Tennenholtz, M.: Some tractable combinatorial auctions. In: AAAI/IAAI, pp. 98–103 (2000)
7. Milgrom, P.: Putting Auction Theory to Work. Cambridge University Press, Cambridge (2004)
8. Cavallo, R.: Optimal decision-making with minimal waste: strategyproof redistribution of VCG payments. In: AAMAS, pp. 882–889 (2006)
9. Clarke, E.H.: Multipart pricing of public goods. Public Choice 11, 17–33 (1971)
10. Jackson, D.: Automating first-order relational logic. In: SIGSOFT FSE, pp. 130–139 (2000)
11. Jackson, D.: Alloy: A lightweight object modelling notation. ACM Trans. Softw. Eng. Methodol. 11(2), 256–290 (2002), `http://alloy.mit.edu`
12. Huth, M.R.A., Ryan, M.D.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, Cambridge, England (2000)
13. Vickrey, W.: Counter-speculation, auctions, and competitive sealed tenders. Journal of Finance 41(33), 8–37 (1961)
14. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial optimization. Wiley, 605 Third Avenue, New York, US (1998)
15. Nisan, N.: Bidding and allocation in combinatorial auctions. In: EC '00. Proceedings of the 2nd ACM conference on Electronic commerce, pp. 1–12. ACM Press, New York (2000)
16. Ausubel, L.M., Milgrom, P.: The lovely but lonely vickrey auction. In: Cramton, P., et al. (eds.) Combinatorial Auctions, MIT Press, Cambridge (2006)
17. Tadjouddine, E.M., Guerin, F.: Verifying equilibria in games of complete and perfect information represented by presburger formulas. In: Proceedings of the 2006 LOFT Conference, Liverpool, UK, pp. 219–226 (July 2006)

Agent Environment and Knowledge in Distributed Join Calculus

Sławomir P. Maludziński and Grzegorz Dobrowolski

Department of Computer Science, AGH University of Science and Technology
smaludzi@student.agh.edu.pl, grzela@agh.edu.pl

Abstract. Distributed join calculus can be used to formally reason about concurrent systems. This paper introduces notion of an environment where agents interact. Based on the introduced formalisms a notion of knowledge which is exchanged by facts and common to all agents is presented. Introduction of the environment and knowledge extends number of agent system which can be expressed using join calculus algebra.

1 Introduction

Several attempts had been taken to formally specify multi-agent systems (MAS). Different methods are used for this purpose. Starting from temporal logic, through abstract state machines and variants of the π-calculus [SW01] [FGL$^+$96]. Current works do not present any formal definition of an environment which can be sensed and affected by agents. This article tries to fill this gap. Based on messages, which are common to all agents, it is possible to define facts which are basis of knowledge representation.

2 The Join Calculus

Distributed join calculus [FGL$^+$96] is an asynchronous variant of Milner's π-calculus with locality and static scoping rules. It allows expressing mobile agents moving between physical sites. Agents are not only programs but core images of running processes with communication capabilities. Thanks to notion of location, which resides on a physical site, one location can be moved atomically to another site (or to constitute a sub-location of a given one). Every agent is specified as a set of processes which define its functionality. Behaviour of an agent includes asynchronous emission of a message (communication), migration to other location (mobility). Agents defined in this calculus are reactive, that is respond to messages which are sent by other agents. Specific actions are encoded as processes.

3 Environment and Knowledge in Join Calculus

We extend the basic join calculus with methods which make agents proactive, that is they can observe surrounding environment. Also, methods are given which make it feasible for agents to reason about the environment. This way we are able to express knowledge and reasoning operations formally.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 298–300, 2007.

3.1 Operational Semantics

Figure 1 expands asynchronous core defined in [FGL+96]. A process can perform two additional operations, $f\langle x\langle\tilde{v}\rangle\rangle$ and $\overline{f}\langle x\langle\tilde{v}\rangle\rangle$, these denote an asynchronous emission of a message to the environment and asynchronous removal of a message from the environment, respectively. Definition D has been altered with newer reaction rule $J|F \rhd P$. Join pattern consists of one or more messages to trigger a guarder process and possibly zero or more messages which have been sent to an environment. This definition seems to be similar to the already presented in the asynchronous core. Its chemical rule is different, though. An attentive reader will notice that constructs of the form $F|F \rhd P$ are prohibited in our definition. Introduction of at least one J asynchronous message sent to trigger a process is to make an agent decide when it wants to interact with an environment. Otherwise such a rule would be triggered repeatedly.

$$
\begin{aligned}
P &::= x\langle\tilde{v}\rangle \mid \mathbf{def}\ D\ \mathbf{in}\ P \mid P|P \mid f\langle x\langle\tilde{v}\rangle\rangle \mid \overline{f}\langle x\langle\tilde{v}\rangle\rangle \mid \mathbf{0} \\
D &::= J|F \rhd P \mid D \wedge D \mid \mathbf{T} \\
J &::= x\langle v\rangle \mid J|J \\
F &::= f\langle x\langle\tilde{v}\rangle\rangle \mid F|F \mid \mathbf{T}
\end{aligned}
$$

Fig. 1. Join calculus core extended by environment

3.2 Abstract Machine Modifications

Changes made in the operational semantics must be reflected by modifications of the chemical abstract machine. Figure 2 presents definitions of new reduction rule. Messages sent are consumed, and newer process P is created. However, messages sent to the environment are left in it unchanged. Any modifications of these messages may be done using processes actions f and $\overline{f}$. In comparison to join calculus chemical rules two additional reduction rules were added. Rules **str-add** and **str-rem** define operations performed onto chemical abstract machine. The most important, however, is the alternation of **red** reduction rule. Its newer semantics imposes explicit addition and removal of facts from the environment.

3.3 Knowledge

Environment which is common to all agents gives us a chance to define variables which will be shared by all agents. The simplest example is a reference to a Naming Service which will be used to discover agents between themselves. Other values may be communicated in the system. Not only primitive ones, but also computed by specialised agents. Presence of facts in the environment and rules which operate on them constitutes framework for reasoning.

$$
\begin{array}{lrcll}
\textbf{str-join} & \vdash P_1 | P_2 & \rightleftharpoons & \vdash P_1, P_2 & \\
\textbf{str-null} & \vdash \mathbf{0} & \rightleftharpoons & \vdash & \\
\textbf{str-and} & D_1 \wedge D_2 \vdash & \rightleftharpoons & D_1, D_2 \vdash & \\
\textbf{str-nodef} & \mathbf{T} \vdash & \rightleftharpoons & \vdash & \\
\textbf{str-def} & \vdash \textbf{def}\ D\ \textbf{in}\ P & \rightleftharpoons & D\sigma_{dv} \vdash P\sigma_{dv} & \mathrm{range}(\sigma_{dv})\mathrm{fresh} \\
\textbf{str-add} & \vdash f\langle x\langle \tilde{v}\rangle\rangle & \rightleftharpoons & \vdash f\langle x\langle \tilde{v}\rangle\rangle\sigma_{rv} & \\
\textbf{str-rem} & \vdash \overline{f}\langle x\langle \tilde{v}\rangle\rangle & \rightleftharpoons & \vdash & \\
\textbf{red} & J|F \rhd P \vdash J\sigma_{rv}, F\sigma_{rv} & \longrightarrow & J|F \rhd P \vdash P\sigma_{rv}, F\sigma_{rv} &
\end{array}
$$

Fig. 2. Join calculus chemical rules for calculus with environment

3.4 Reduction to Asynchronous Core

The proposed extension of the distributed join calculus may be reduced to the asynchronous core. Due to lack of space precise details are not shown in this short article. Reduction of the calculus with an environment to the asynchronous join calculus core preserves all properties defined by the join calculus. Observational equivalences remain same; we may reason about program equivalences. May testing equivalence and fair testing equivalence will be defined in the same way.

4 Summary

Multi-agent systems may be regarded as concurrent systems and thus can be described formally. This article outlines how agent environment and knowledge can be described in a process calculus. Authors of this paper hope that such a description will be useful to specify, synthetize and verify such systems.

References

[BB90] Berry, G., Boudol, G.: The chemical abstract machine. In: POPL '90. Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 81–94. ACM Press, New York (1990)

[FG96] Fournet, C., Gonthier, G.: The reflexive cham and the join-calculus. In: POPL, pp. 372–385 (1996)

[FGL+96] Fournet, C., Gonthier, G., Lévy, J.-J., Maranget, L., Rémy, D.: A calculus of mobile agents. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, Springer, Heidelberg (1996)

[SW01] Sangiorgi, D., Walker, D.: Pi-calculus: A theory of mobile processes. Cambridge University Press, New York, NY, USA (2001)

Agent-Based Architecture of Intelligent Distance Learning System

Mikhail Pashkin

St.Petersburg Institute for Informatics and Automation
of the Russian Academy of Sciences
39, 14-th Line, 199178, St.-Petersburg, Russia
michael@iias.spb.su
cais.iias.spb.su

Abstract. During several recent decades a lot of research projects have been done in the area of intelligent distance learning systems. Such systems became wide spread in different educational and industrial organizations due to rapid evolution of network engineering and Internet. The physical presence of teacher and students in one class is not required due to usage of such systems. Usually, such systems have distributed architecture. This was a major motivation to make a research to analyze possibility of multiagent technologies usage in intelligent distance learning systems. The paper presents the developed architecture of agent community for such systems and a research prototype allowing to model agents' interaction for user interface control based on several pedagogical strategies.

Keywords: Multiagent system, profiling, intelligent distance learning system.

1 Introduction

Currently e-Learning has become very popular in many educational and industrial organizations due to the facts that many people aim to get education in different areas, student population is very diverse and geographically dispersed, rapid growth of network and multimedia technologies opens new opportunities to learning content presentations and other. Some of well-known problems in e-Learning are: (i) students can get disorientated in the course material, (ii) students may lose their motivation, (iii) individually tailored instruction involving one-on-one attention is too costly, logistically challenging, etc. Hence, the task of distance learning system (DLS) implementation for the above problem solving is actual [1].

General DLS includes four modules: domain model ("what to learn"), pedagogical model ("how to learn"), student model ("whom to learn"), and interface. Ontologies were adapted by many researchers in the area of DLS as a formalism for domain knowledge representation. Usage of pedagogical strategies allows to interfere into the process of learning to increase the level of students' mastery (e.g., to present tips and hint, to change amount of learning content, etc.). User profiles are widely used for student modeling. In the presented research the following features of students have been used: learning style based on Felder and Silverman model [2] and estimation of

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 301 – 303, 2007.

skill based on results of test and navigation in graphic user interface. In the prototype, agents control user interface using pedagogical strategies and user profiles.

The paper briefly presents results of the on-going research devoted to application of multiagent technologies to intelligent DLS design. Section 2 presents the developed agent community. Section 3 gives a brief conclusion and direction for further research activities.

2 Agent Community for Intelligent Distance Learning System

The following requirements have been considered as important for intelligent DLS: (i) *scalability*: new learning courses can appear and new methods for skill estimation and data analysis can be developed (ii) *customization*: learning content has to be presented to students in the most suitable way and has to have adequate amount to guarantee level of skill; (iii) *distributed architecture*: the system has to be able to serve geographically dispersal students.

Based on the analysis of results of fulfilled projects using agent technologies in the area of DLSs [3] and taking into account methodological recommendations for multiagent systems development, architecture of agent community has been designed (Fig. 1).

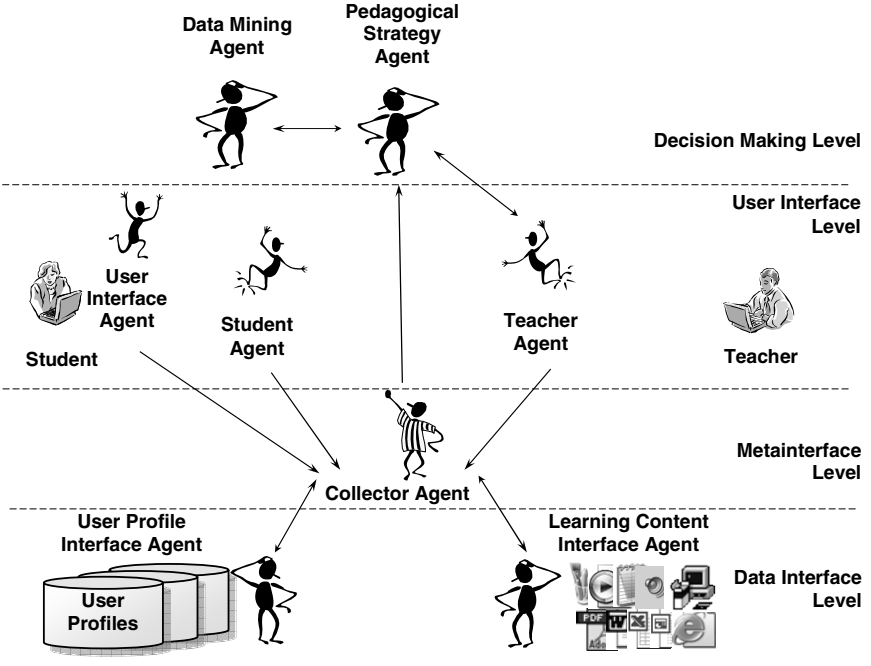


Fig. 1. Developed architecture of agent community of intelligent distance learning system

The proposed agents are divided into the three groups: (i) decision making agents that analyze data and require performance of actions from the side of DLS:

Pedagogical Strategy Agent – interprets raw data received from Collector Agent and make decision which rule from pedagogical strategy ontology should be executed; and *Data Mining Agent* – analyses of raw data to define association rules between student behavior and student mastery and cluster students using decision trees and neural networks, (ii) interface function agents that are responsible for learning content presentation: *User Profile Interface Agent* – reads and modifies user profile; *Learning Content Interface Agent* – reads learning content description for presentation to a student and provides physical access to required elements; *Collector Agent* – accumulates auxiliary data from several agents; and *User Interface Agents* – reflects of requirements to learning content presentation for a certain student from the side of DLS and (iii) human agents that model and represent of the human and perform tasks on behalf of them: *Student Agents* – represents and models a student; and *Teacher Agents* – represents and models a tutor.

The major task for the multi-agent research prototype of DLS development is to model active interface for presentation of learning content depending on student mastery and learning style. A software framework JADE [4] has been chosen for implementation of the proposed agents. Problem-oriented functions and user interfaces have been implemented using Java programming language.

3 Conclusion

The paper discusses a multi-agent architecture for intelligent distance learning system implementation. Based on the analysis of results of fulfilled projects using agent technologies in this area and taking into account methodological recommendations for multiagent systems development 8 types of agents have been proposed.

In the framework of research prototype a possibility to control user interface from the server side based on several pedagogical strategies, accumulated raw data and learning style has been verified and showed. A set of pro-active agent functions has been also implemented and tested in the research prototype.

Acknowledgments. The paper is due to research projects supported by grant # 07-01-00334 of the Russian Foundation for Basic Research.

References

1. Vasilakos, T., et al.: Computational Intelligence in Web-Based Education: A Tutorial. Journal of Interactive Learning Research 15(4), 299–318 (2004)
2. Felder, R., Silverman, L.: Learning and Teaching Styles in Engineering Education. Engineering Education 78(7), 674–681 (1988)
3. Reyes, R., Sison, R.: A Case-Based Reasoning Approach to an Internet Agent-based Tutoring System. In: AIED 2001. Proceedings of the World Conference on Artificial Intelligence in Education, San Antonio, USA, pp. 122–129 (2001)
4. JADE: Java Agent DEvelopment Framework, URL: http://jade.tilab.com

An Architecture and Framework for Agent-Based Web Applications

Alexander Pokahr and Lars Braubach

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
{pokahr,braubach}@informatik.uni-hamburg.de

Abstract. The construction of web applications is a complex task as different kinds of technologies need to be integrated. To ease the task of developing web applications many different web frameworks have been conceived. These frameworks aim at providing support for recurring and tedious development tasks and address complexity by separating the basic concerns of applications. Most of the currently available web frameworks adhere to the widely accepted Model 2 design pattern that targets a clean separation of model, view and controller parts of an application in the sense of the model view controller (MVC) pattern. In this paper it is shown how the basic Model 2 archiecture can be adapted for enabling its usage in combination with business logic realized with agent technology. Besides the architecture itself additionally its realization within the Jadex Webbridge framework is sketched.

1 Introduction

The popularity of web applications is steadily increasing for which one important reason is that they can be accessed via browsers in a standardized way. In this regard they facilitate the execution of arbitrary applications without the need for installing or updating software components. These characteristics make web applications desirable even for more advanced and complex business tasks. Agent technology has already been used in many research and industrial projects for building enterprise scale applications [4,2], but only few works exist that aim at a systematic integration of agent and web technology allowing to easily build agent-based web applications. Hence, in the following an architecture and a corresponding framework for the efficient development of agent-based web applications are presented.

2 Architecture

The main aim of the approach consists in separating the agent-specific parts of an application from the web-specific parts allowing web and business logic developers to focus on their individual competency respectively. Foundation of the proposed architecture is the widely used and accepted Model 2 design pattern [3]. The pattern proposes a separation of concerns, whereby each of the three

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 304–306, 2007.

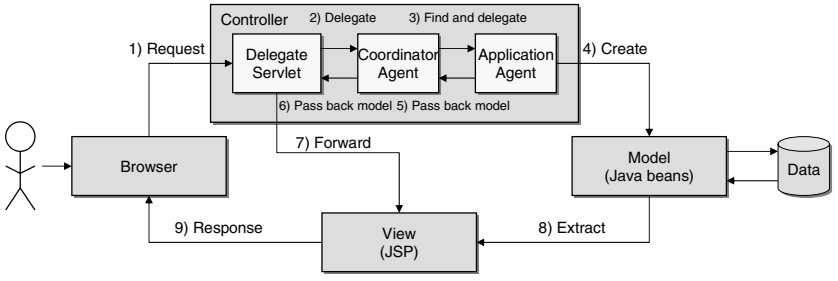


Fig. 1. Agent-based Model 2 architecture

proposed aspects plays a fundamentally different role. The *model* represents the domain-specific representation of the data on which the application operates. It is used by the *view* which has the purpose to render the data in a user-friendly manner. In between the *controller* serves as a connector that translates interactions with the view into actions to be performed on the model.

In an agent-based web application, the agents are responsible for the execution of the application logic. In the traditional Model 2 architecture, the application logic is executed by the controller, which is realized as a Java servlet. To achieve the seamless integration of agents with the web, the Model 2 architecture is extended as shown in Fig. 1 to allow for the execution of agent behavior inside the controller. For this purpose the controller is split into three different entities. The *delegate* servlet forwards the browser request to the agent layer and renders the result by redirecting to a chosen JSP. On the agent side a dedicated *coordinator* agent is responsible for finding or creating a suitable *application* agent that is capable of handling the request. The result is then passed back from the application agent to the coordinator which communicates it back to the delegate servlet.

3 Framework

The agentified Model 2 architecture presented above has been realized in a generic software framework based on the Jadex BDI (belief-desire-intention) agent system [1].[1] This new framework, called *Jadex Webbridge*, enables application developers focussing on the three core aspects of an application, i.e. the application logic using agents, visualization via JSP pages, and the domain data utilizing Java objects.

In the Webbridge framework the delegate servlet and the coordinator agent are responsible for mediating between these elements. They have been realized as reusable components within the framework and offer clear configuration and extension points. From the perspective of an application developer the main task consists in configuring the application settings via the normal web.xml file and additionally in developing the application agents for handling the web requests. The latter task is simplified by a generic agent module, called web interaction

[1] `http://jadex.sourceforge.net`

capability, which can be included by the developer into application agents and handles all communication aspects with the coordinator behind the scenes.

The generic coordinator automatically forwards Web requests as agent messages to an application agent. The *web interaction capability* is included into an application agent and automatically handles request messages sent by the coordinator. For each message a goal of type web_request is created, which has to be handled by custom application plans. The result of the goal processing is automatically communicated back to the coordinator. From the viewpoint of application developers, the web interaction capability converts web requests into goals that belong to the application agent. Therefore, the details of the web request handling are abstracted away from agent programmers allowing them to focus on the behavior of the application agent. This behavior can be built taking the intentional stance and applying the BDI-specific mentalistic notions such as goals and plans. Request processing can of course involve interactions with other agents if appropriate.

4 Conclusion

In this paper, an architecture and a corresponding framework have been presented that allow combining agent and web technology. The proposed architecture extends the well known Model 2 design pattern for web applications and introduces a decomposition of the controller part into a delegate servlet, a coordinator agent and application agents. Supporting the separation of concerns established by Model 2, the architecture further separates the web layer from the agent layer, thereby providing a solid foundation for combining agent-based application functionality and web-based user interaction.

To ease the development of applications following the proposed architecture, the Jadex Webbridge framework has been presented, which provides ready-to-use and extensible functionalities realizing the delegate servlet and the coordinator agent. Additionally, a web interaction module (capability) is provided that encapsulates the generic functionalities needed by application agents. It abstracts away details of message-based communication and reduces the task of the application agent developer to writing plans for the domain logic of pursuing goals, which correspond to web requests.

References

1. L. Braubach, A. Pokahr, and W. Lamersdorf. Jadex: A BDI Agent System Combining Middleware and Reasoning. In *Software Agent-Based Applications, Platforms and Development Kits*, pages 143–168. Birkhäuser, 2005.
2. J. Castro, M. Kolp, and J. Mylopoulos. Developing agent-oriented information systems for the enterprise. In *Proc. of the 2nd Int. Conf. on Enterprise Information Systems (ICEIS 2000)*, pages 9–24. ICEIS Secretariat, 2000.
3. N. Ford. *Art of Java Web development: Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, WebWorks*. Manning Publications, 2003.
4. N. R. Jennings and M. J. Wooldridge. *Agent Technology - Foundations, Applications and Markets*. Springer, 1998.

Closing the Gap Between Organizational Models and Multi-Agent System Deployment

Michael Köhler and Matthias Wester-Ebbinghaus

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D–22527 Hamburg
{koehler,wester}@informatik.uni-hamburg.de

Abstract. Multi-agent system research deals with organization theoretical concepts in order to analyse and control supra-individual phenomena. However, there exists a conceptual gap between organizational specifications and their multi-agent implementation. We address this problem by presenting an integrated approach for the specification and deployment of organizational models based on Petri nets.

1 Introduction

Organization theoretical concepts serve to analyse and control phenomena in multi-agent systems that carry a *supra-individual* character (we provide a detailed overview of recent and current work in [1]). The corresponding increase in complexity calls for an exact conceptualisation. Formal notations are qualified for precisely specifying system models and additionally offer the possibility of verifying these models with respect to certain properties. But they most often do not lend themselves to direct implementation. We address this problem by presenting an integrated approach for the specification and deployment of organizational models based on Petri nets. The operational semantics of Petri nets allows them to be directly utilized in a software implementation. We present a formal Petri net model of organizations in Sect. 2. In Sect. 3 we demonstrate how the formal specifications may directly be operationalised inside a multi-agent deployment. We conclude our results in Sect. 4.

2 Formal Model of Organizations

In [2] we introduce our formal Petri net based model of organizations. Figure 1 shows an example. According to the basic requirements for organizational specifications postulated by Mintzberg ([3]), this organization net captures division of labour and coordination of the corresponding distributed workflows. Each transition models a task and each place models a role profile. A task may introduce multiple roles between which the labour is divided and each new role has itself to be implemented by a task. In order to coordinate the distributed execution of a task, each transition is assigned a service net that describes the corresponding interaction scenario between the included roles in detail.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 307–309, 2007.

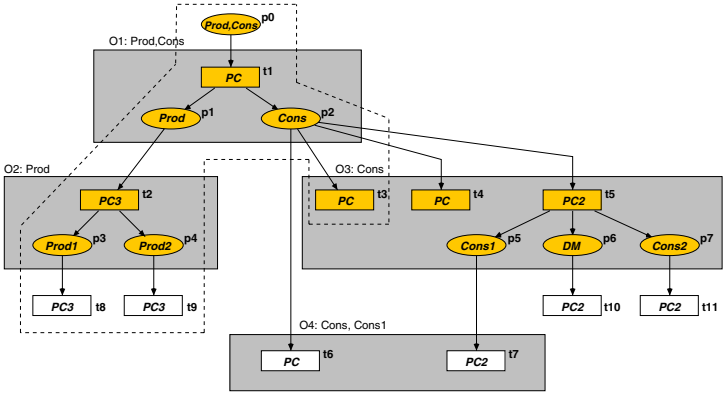


Fig. 1. Producer/Consumer organization net

Tasks and related role profiles are clustered into organizational positions. These positions roughly correspond to positions in real-world organizations. Each position is responsible for some tasks and is allowed to delegate parts of these tasks to other positions. There are different alternatives concerning delegation paths. In order to actually carry out services, teams as temporary structures are formed. One possible team is marked by the dashed lines in Fig. 1.

As our approach is grounded in the theory of Petri nets, various formal properties and requirements (above all, well-formedness) are associated with organizational models and can be formally verified.

3 Multi-Agent System Deployment

An organization net implicates a multi-agent system design including different kinds of agents like it is shown in Fig. 2 (a). Each organizational position is mapped onto a *position agent*. Position agents are directly derived from the structure of an organization net and can be considered as administrative agents that are *owned* by the organization. *Member agents* are those that actually carry out actions and make decisions. But they must connect to position agents and use them as gateways to the organization. By this means, the organization ensures inter-operability between its agents and holds a strong degree of monitoring and control over its functioning.

Besides guiding the coarse design of corresponding multi-agent systems, organization nets may also be directly utilized inside the implementation due to the operational semantics of Petri nets. So called deployment versions of organization nets are used for different purposes. Figure 2 (b) shows a fragment of a deployed organization net for the purpose of team formation. It reveals additional inscriptions compared to an ordinary organization net and additional net components for team formation calls. Deployed organization nets (or just fragments of them) are distributed among the position agents, which use them

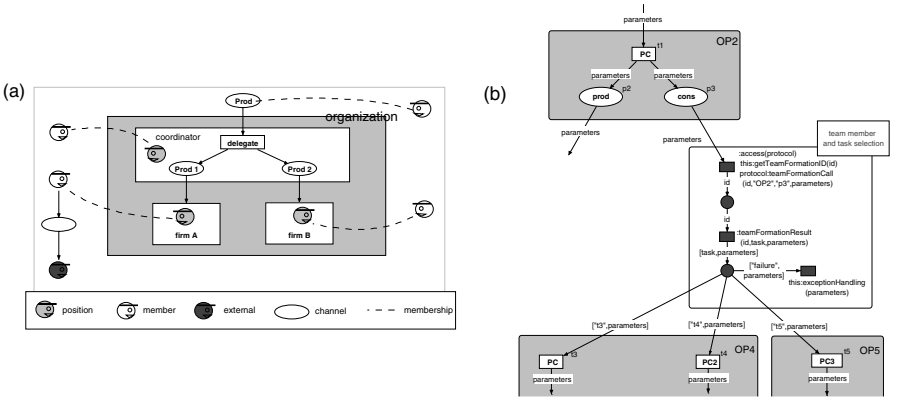


Fig. 2. MAS deployment: (a) system design (b) deployment version for team formation

to steer and monitor organization-related processes like team formation or team plan execution.

4 Outlook

We presented an approach to use Petri nets to both specify and deploy organizational models that especially focuses on closing the conceptual gap between specification and implementation. Turning to future work, we consider the result of the paper at hand as one step into the direction of our vision of organization-oriented software engineering. In [1] we present a software architecture based on layered organizational units. As our deployment approach presented in the paper at hand is explicitly and directly derived from our formal model we arrive at a theoretically grounded and exactly conceptualised basis for organizational units.

References

1. Wester-Ebbinghaus, M., Moldt, D., Reese, C., Markwardt, K.: Towards organization–oriented software engineering. In: Züllighoven, H. (ed.) Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings, LNI, GI (2007)
2. Köhler, M.: Formalising multi-agent organisations. In: Burkhard, H.D., Czaja, L., Lindemann, G., Skowron, A. (eds.) Proceedings of the International Workshop on Concurrency, Specification, and Programming (CS&P 2006) (2006)
3. Mintzberg, H.: Structure in Fives: Designing Effective Organizations. Prentice-Hall, Englewood Cliffs (1983)

Clustering Techniques in Automated Purchase Negotiations*

Miguel A. Lopez-Carmona, Juan R. Velasco, and Ivan Marsa-Maestre

Department of Automatica, University of Alcala
Edificio Politecnico, Ctra. NII, km. 33.600
28871 Alcala de Henares (Madrid), Spain
Tel.: +34 918856673, Fax:+34 918856641
{miguelangel.lopez,juanramon.velasco,ivan.marsa}@uah.es

Abstract. In this paper we propose a modification to a fuzzy constraint based framework for automated purchase negotiations in competitive trading environments. The goal of this work is to improve the performance of the negotiation processes in terms of computation time and joint utility. This modification is based on the use of clustering techniques in the seller's decision mechanisms.

1 Introduction

Automated negotiation is an important challenge in the Multi-Agent Systems community which has been covered from different areas such as game theory and distributed artificial intelligence [1]. In our previous work [2], we present a fuzzy contraint based model for automated purchase negotiations, and show how with our approach the negotiation processes are more efficient than with previous approaches [3] where mainly positional bargaining is used. The negotiation model uses fuzzy constraints [3] to capture requirements and to express proposals, and the proposed interaction protocol is a dialogue game protocol [4]. We propose a modification to the seller's decision mechanisms in order to improve the negotiation process, i.e. we expect to increase the agents' utilities obtained by the agreement achieved, and diminish the computational overhead.

2 Preliminaries

Let A_b and A_s represent a buyer and a seller agent, a negotiation thread is a finite sequence of proposals from one agent to the other. During the negotiation stage A_b utters *purchase requirements* $\lambda_{A_b} = \bigcap \{R_i^c | i = 1...m\}$. A purchase requirement is a purchase proposal which is formed by a set of crisp constraints extracted from a set of fuzzy constraints which describe the buyer's preferences regarding the attributes of the products. On the other hand, A_s may respond to

* This work has been supported by the Spanish Ministry of Education and Science grant TSI2005-07384-C03-03.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 310–312, 2007.

a buyer agent in three different ways: rejecting the proposal, offering a product which satisfy the purchase requirement, and suggesting the relaxation of the purchase requirement. As we demonstrate in [2], the use of relaxation requirements improves the negotiation processes, so a seller agent should implement a mechanism to generate relaxation requirements. In [2] we implement this mechanism as a two step process.

Generate Potential Sale Offers makes a selection of products which the seller agent considers as good candidates for a sale offer. We have identified two main aspects when making this selection: the *local utility* u_j, and the *viability*, which depends on the similarity between a product p_j and the purchase requirement λ_{A_b}. The *viability* estimates the validity of a product as a future sale offer. A seller agent may give more or less importance to each aspect by means of the $\beta \in [0, 1]$ parameter. The $prefer$ function estimates the goodness of a potential sale offer in terms of *utility* and *viability*:

$$prefer(p_j) = \beta * u_j + (1 - \beta) * \hat{viability}(p_j, \lambda_{A_b})$$

$$prefer(p_j) = \beta * u_j + (1 - \beta) * (1 - sqrt(\sum_{i=1}^{n} (\hat{dist}(a_{ji}, \lambda^i_{A_b}))^2/n))$$

where $\hat{dist}(a_{ji}, \lambda^i_{A_b})$ represents a per-attribute estimate of distance. Once the $prefer$ function has been computed for all the products in the catalogue, those products with a value exceeding a threshold are selected. This is the set S_p.

Generate Relax Requirement generates the relax requirement $\rho_{A_s} = (r_1, ..., r_i, ...)$ to submit to the buyer agent, where $r_i = 1$ if constraint R^{f}_i in λ_{A_b} has not been satisfied for any product in S_p, and otherwise $r_i = 0$.

3 Contributions

We have detected two main problems with the previous approach: firstly, this mechanism is very time consuming for large catalogues of products; and second, the similarity (distance) calculus may distract the seller agent when generating the relaxation proposals. It must be noted that a $prefer$ value is computed for each product in the catalogue and for each negotiation round, and so the *Generate Potential Sale Offers* mechanism may incur in a computational overhead in the case of large catalogues. On the other hand, the *viability* estimate of a product as a sale offer is based on a distance measure which is very sensitive to the β parameter. Our aim is to provide a new version of the *Generate Potential Sale Offers* mechanism which incur in a less computational overhead, and at the same time make the mechanism less sensitive to the β parameter. The main idea is to apply clustering techniques to the seller's product catalogue in order to improve the performance of the seller's generation of potential sale offers mechanism, which is a key part in the generation of the relaxation requirements. So, we propose the following modifications in order to compute the *prefer* value of a product for the *Generate Potential Sale Offers* mechanism of the seller agent.

1) Before entering a negotiation dialogue the seller agent applies a clustering algorithm to the catalogue of products S. It generates a set of product representatives $Rep(S)$. Depending on the characteristics of the catalogue this set may be considerably smaller than the catalogue of products.

2) The $prefer$ value is computed for each product and each negotiation round in the following way:

- For a given product p_j the distance estimate is computed between the purchase requirement received and the representative of the product p_j, not the product itself. If we compute at the begining of each round the distances from the purchase requirement to the products in $Rep(S)$, we will not need to repeat this calculations for all the products in the catalogue, and so it may improve the computation time. On the other hand, the variations of the distance estimates will be smaller because the references will be the representatives, not the products.
- For a given product p_j the local utility u_j is computed as in the original mechanism, i.e. considering the utility of p_j and not the utility of the representative.

4 Conclusions and Future Work

This paper presents a novel approach for the seller agent's decision mechanism which is in charge of the generation of relaxation requirements. Our proposal implies the modification of the mechanism which generates the potential sale offers in each negotiation round, and it is based on the use of clustering techniques. With our approach we expect to improve the results of the negotiation processes in two aspects, robustness and computation time.

As future work we propose to make an exhaustive analysis of different clustering techniques that could be applied in order to test the negotiation framework with the proposed modifications.

References

1. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter. MIT Press, Cambridge MA, USA (1994)
2. Lopez-Carmona, M.A., Velasco, J.R.: A fuzzy constraint based model for automated purchase negotiations. In: TADA/AMEC 2006. LNCS (LNAI), vol. 4452, pp. 234–247. Springer, Heidelberg (2007)
3. Luo, X., Jennings, N.R., Shadbolt, N., Ho-Fung-Leung, Lee, J.H.M.: A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. Artificial Intelligence 148(1-2), 53–102 (2003)
4. McBurney, P., Euk, R.M.V., Parsons, S., Amgoud, L.: A dialogue game protocol for agent purchase negotiations. Journal of Autonomous Agents and Multi-Agent Systems 7(3), 235–273 (2003)

Cooperative CBR System for Sharing Student Models in Cooperative Intelligent Tutoring Systems *

Carolina González [1,2], Juan C. Burguillo[1], and Martín Llamas[1]

[1] Departamento de Telemática, Universidad de Vigo, Spain
[2] Departamento de Sistemas, Universidad del Cauca, Colombia
{cgonzals,jrial,martin}@det.uvigo.es}

Abstract. Cooperation is the fundamental characteristic of multi-agent systems where the overall system exhibits significantly greater functionality than the individual components. In this paper, we propose a P2P cooperation framework for multiple Case-based Reasoning (CBR) agents to handle the student models initialization problem in Intelligent Tutoring Systems (ITSs). Our work shows that using the "committee" collaboration policy the agents can obtain better results that working alone, allowing to enhance the system overall performance.

Keywords: Cooperative CBR, Multi-agent Systems, Student Model, Intelligent Tutoring Systems.

1 Introduction

The initialization of a student model (ISM) [1] is an important function of the student modeler of Intelligent Tutoring Systems (ITSs) [2] Although a lot of research work has focused on the identification of efficient methods for updating the student model, the process of the initialization has often been neglected or it has been dealt using trivial techniques. ISM is of great importance because it seems unreasonable to assume that every student starts up with the same knowledge and misconceptions about the domain being taught. In this sense, a fast und efficient initialization process is necessary. Otherwise the ITSs may lose its credibility in the first interaction with the student. In this paper, we introduce a cooperative framework to handle the student models initialization problem in ITSs. A committee of agents using case-based reasoning (CBR) [3] carries out the initialization und updating process for each student taking into account information that originates from other student's performance while using the system. The CBR methodology offers the multi-agent system the capability of autonomously learn from experience. The agent cooperation imposed by our system improves its performance through the sharing of multiple experiences.

2 Cooperative CBR Multi-agent System for ISM

Our approach allows taking advantage of past experiences of a student or student groups. Each student is described by a case containing domain independent information (DII) and

* The work was funded by the project MetaLearn (TIN2004-08367-C02-01).

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 313 – 315, 2007.

domain specific information (DSI). The DII includes student's personal information, background, experience, goals, and preferences of learning style. The DSI contains student competence levels for each concept node and each unit in the content tree. Figure 1 shows our system.

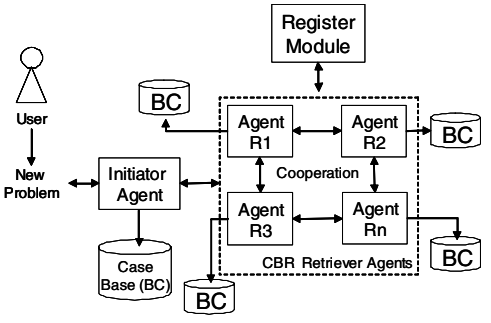


Fig. 1. Cooperative CBR multi-agent system

The multi-agent system [4] CBR is composed on *n* agents, where each agent has its own case base. In a formal definition, a case is an ordered pair (P, S) where *P* is the problem space, and *S* is the solution space. The cooperation process includes the following steps and resembles the Contract Protocol [5] of multi-agent systems:

- An initiator agent observes student's actions (new problem) on his local database and determines its initial profile. The agents apply the Lazy Induction Description algorithm (LID) [6], in order to determine the more relevant features of the problem.
- For each relevant feature, the initiator agent sends a request to a "committee" of peers. This request is broadcasted to all peer agents composing the committee.
- Each agent contacted, denoted as the retriever agent, searches students with similar characteristics related with the received request, and sends the result to the initiator agent. The research and retrieval is done using CBR approach in order to reuse the solution for future similar problems, and to improve the case search process.
- The initiator agent handles the results received from the retriever agents. The result obtained contains recommended cases lists.
- The initiator agent applies a *voting scheme* in order to rank recommended cases. The *K* highly ranked recommended case will be proposed to be applied to the actual problem.

Each agent in the MAS carries out the CBR cycle, in order to select the more relevant cases to the actual problem. Each CBR agent implements the next phases:

- *Search phase:* For each relevant feature, the retriever agent applies a CBR cycle. The committee compares the target feature with one of the case in the agent's case base.

- *Reuse phase:* Having all similar cases, the system estimates the degree of similarity of the searched case by using fixed values in a threshold. If the degree of similarity is higher than the fixed threshold value, then the case is retained.
- *Revision Phase:* The new case is evaluated by the initiator agent according to the student behavior faced with the proposed solution. Each agent will learn by adding in its case base new well evaluated cases or by updating existing cases. If there is no solution found in the search phase, the new formed case will be added to the case base of the initiator agent. But, if the case exists already in the case base for the same agent, then the case specifically, the evaluation part will be updated with the new values.

3 Discussion and Conclusion

This paper describes a work in progress that aims at finding the most relevant cases from the appropriate agents. The approach is based on agents collaborating in order to provide good solutions that allow improving the students' performance during the learning process. The proposed approach is simple and provides efficient cooperation framework for retrieval of distributed case-bases. Currently, we are working on implementing the first version of this system. An important aspect to handle is the committee formation: how to learn to form the optimal committee for each new problem. We will carry out a simulation process with two different sceneries for each new problem:

- the request is broadcasted to all available agents (in our case n agents)
- the request is sent exclusively to a set of agents leaders, that manage groups of agents. This architecture for P2P communications resembles Kazaa [7].

References

1. Tsiriga, V., Virvou, M.: A Framework for the Initialization of Student Models in Web-based Intelligent Tutoring Systems. User Modeling and User-Adapted Interaction 14(4), 289–316 (2004)
2. Beck, J., Stern, M., Haugsjaa, E.: Applications of AI in Education. ACM Student Magazine (1996)
3. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1), 39–59 (1994)
4. Wooldridge, M.: Introduction to Multi Agent Systems. John Wiley and Sons, Chichester (2002)
5. FIPA Contract Net Interaction Protocol Specification: FIPA TC Communication. Document No. SC00029H (2002)
6. Armengol, E., Plaza, E.: Lazy Induction of Descriptions for Relational Case-Based Learning. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, Springer, Heidelberg (2001)
7. Kazaa peer-to-peer file sharing service, accessible at www.kazaa.com

Decision Making System: Agent Diagnosing Child Care Diseases

Vijay Kumar Mago[1] , M. Syamala Devi[2], and Ravinder Mehta[3]

[1] Department of Computer Science, DAV College, Jalandhar, India
v_mago@yahoo.com
[2] Department of Computer Science and Applications, Panjab University, Chandigarh, India
syamala@pu.ac.in
[3] Consultant Pediatrician, Vijayanand Diagnostic Center, Ludhiana, India

Abstract. The use of Multi-agent systems (MAS) in healthcare domain has been widely recognized. But, due to uncertainty in medical domain, it is difficult to decide the appropriate agent with whom to collaborate in order to find the proper diagnosis for a given set of sign-symptoms. To overcome this problem, a Bayesian network (BN) based decision making framework is presented. The proposed methodology is applied to an Intelligent Pediatric Agent (IPA), a part of MAS for child care [1].

Keywords: Multi-agent System, Decision making, Bayesian Networks, Child care.

1 Introduction

This paper presents the design and preliminary evaluation of a decision making of an Intelligent Pediatric Agent (IPA) that chooses among super specialist agents (SSA) for tackling childhood diseases. The decision making system is built using Bayesian Network, an inference mechanism to build an uncertain-reasoning system. The IPA is to decide which SSA is to be contacted as per the sign symptoms provided to it by an agent called User Agent (UA). The detailed design was presented in [1]. The following super specialists are under consideration:

- *Endocrinologist:* Deals with diseases of endocrinal glands which secrete hormones
- *Cardiologist:* A specialist who deals with diseases due to malfunctioning of heart.
- *General Surgeon:* A doctor who treats diseases through surgery.
- *Pulmonologist:* A specialist who is required in lung diseases.
- *Gastroenterologist:* A specialist who is required in intestinal and liver diseases.

2 System Design

2.1 Childhood Diseases

Some of the childhood diseases are not diagnosed properly by health care workers in rural India, as discussed in [1]. Using the proposed system, the healthcare practitioner

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 316 – 318, 2007.

is to send the sign-symptoms to the IPA. The IPA is responsible to select the concerned super specialist. A few diseases are taken for construction of BN. These are shown in table 1.

Table 1. Diseases and its description

DISEASE	DESCRIPTION
Asthma	Chest disease, in which there is allergic cough coming in bouts associated with breathlessness and a wheezing sound.
Tuberculosis	Infection caused by Mycobacterium tuberculosis.
Ischemic Heart Disease	Decreased blood supply to heart muscle.
Hiatus Hernia	Upper end of stomach herniates through diaphragmatic opening.
Acid Peptic Disease	Common gastritis/Acidity.
Pneumonia	Inflammation of lung parenchyma.
Heart Failure	Disease in which pumping action of heart is compromised due to various causes.
Diabetes	Metabolic disease in which insulin is decreased resulting in high blood glucose levels.
Hypothyroidism	Hormonal disease in which thyroid-hormone levels decrease
Intestinal obstruction	When there is a block in gut-passage.
Calculi	Stones e.g. kidney stone, gall stone

Some of these diseases are critical and may lead to infant mortality if not taken care.

2.2 Bayesian Network for IPA

We are now illustrating the BN developed for IPA, shown in figure 2. It is being constructed with the help of GeNIe [2], an interactive tool for development of BN.

There are 34 nodes in the BN out of which 11 nodes are dedicated to different diseases to be handled by 5 specialists. For instance, Breathlessness as a clinical

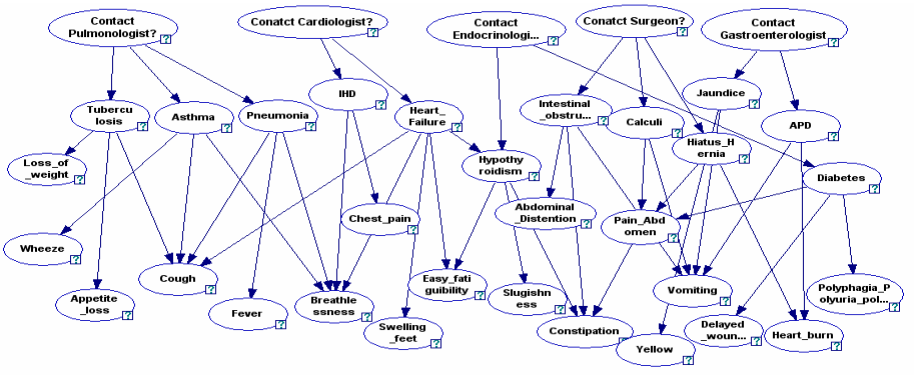


Fig. 2. Bayesian Network for Intelligent Pediatric Agent

symptom may be caused by Asthma, Pneumonia, Heart failure or IHD. The details of these diseases are given in table 1. These diseases are ideally to be tackled by either a Pulmonologist or a Cardiologist.

In the next section the results produced by BN and opinions of a Pediatrician and general doctors have been analyzed.

2.3 Evaluation

We consult a Pediatrician and a General Doctor (GD). Both are given a set of sign-symptoms and requested to rank the specialist, as per their choice for consultation. It is important that first choice of specialist should match with the outcome of BN.

The results indicate that the outcome of BN for IPA and Pediatrician match whereas the selection by a GD posted at rural dispensary is not satisfactory. For instance, we provide sign-symptoms as 'Swelling feet, Easy fatigue' as evidence to IPA. It suggests contacting Cardiologist for reference as preference 1. Same is being suggested by a Pediatrician but the GD prefers to refer the case to Pulmonologist. The procedure has been repeated a number of times, but most of the time the GD's opinion does not match either with IPA or Pediatrician. This suggests that BN for IPA produces results that go well with those of a Pediatrician and also BN has properly encoded the knowledge of a Pediatrician.

3 Conclusion and Scope for Future Work

To make an agent intelligent enough to behave like a pediatrician, for deciding the specialist whenever need arises has been encoded in a BN. The results suggest that BN is a useful technique for selection among agents and can be incorporated in larger MAS.

During the next phase, the BN will be enhanced to include more specialists like Neonatologist, Neurologist etc. and consequently more diseases. This will enable us to integrate it with the work reported in [1].

Acknowledgement. We wish to acknowledge the efforts of Dr. Punam Sekhri, Medical Officer and Dr. Ritu Mehta, Medical Service Provider, posted in rural parts of India for assisting in analyzing the results of BN.

References

1. Mago, V.K., Devi, M.S.: A Multi-agent Medical System for Indian Rural Infant and Child Care. In: Int. Joint Conference on AI, pp. 1396–1401 (2007)
2. http://genie.sis.pitt.edu.

FIPA-Based Interoperable Agent Mobility

Jordi Cucurull, Ramon Martí, Sergi Robles, Joan Borrell, and Guillermo Navarro

Department of Information and Communications Engineering,
Universitat Autònoma de Barcelona,
08193 Bellaterra - Spain
{jcucurull,rmarti,srobles,jborrell,gnavarro}@deic.uab.cat

Abstract. This paper presents a proposal for a flexible agent mobility architecture based on IEEE-FIPA standards and intended to be one of them. This proposal is a first step towards interoperable mobility mechanisms, which are needed for future agent migration between different kinds of platforms. Our proposal is presented as a flexible and robust architecture that has been successfully implemented in the JADE and AgentScape platforms. It is based on an open set of protocols, allowing new protocols and future improvements to be supported. With this proposal we demonstrate that a standard architecture for agent mobility capable of supporting several agent platforms can be defined and implemented.

Keywords: Mobile Agents, interoperability, FIPA, mobility, migration architecture.

1 Introduction

Agent mobility is the agent ability of travelling from one execution environment (platform) to another across a computer network. Their introduction and study during the last decade has opened an interesting research field with new applications [13] and paradigms. Unfortunately, mobile agents have also raised some issues regarding security [12] and interoperability [11], which are still unsolved.

Some contributions to agent mobility are agent platforms like AgentScape Operating System [10], Aglets [9] or D'Agents [7]; or recent works on mobility like the Kalong architecture [2]. These platforms are incompatible between them and do not provide standard agent facilities. For this reason, the IEEE-FIPA organisation created several specifications for agent management [6] and communications [5], and even one for agent mobility [4] deprecated due to a lack of implementations. Also, there is our previous contribution [1], a simple inter-platform mobility service, for the JADE agent platform [8], based on the IEEE-FIPA specifications which is currently used within the whole JADE users community.

This paper presents a new agent migration proposal [3], natural evolution of our previous contribution, based on a multi-protocol software architecture which provides for a flexible agent migration process. This proposal is the first step towards a full interoperable agent migration between different agent platforms by defining the whole migration process, but leaving the final execution of the agent open to future research.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 319–321, 2007.

2 Multi-protocol Based Architecture Proposal

This proposal defines a new agent migration architecture that contributes to create a flexible migration framework supporting any set of migration protocols freely chosen by each agent. In this proposal, communication standards proposed by the IEEE-FIPA organisation are used: the mobile agent conforms to the IEEE-FIPA Agent Management Specification [6], all exchanged messages rely on ACL Messages [5], and IEEE-FIPA Interaction Protocols are used.

The whole migration process is split into five steps contained inside a *Main* protocol (see Figure 1): Pre-Transfer, Transfer, Post-Transfer, Agent Registration, and Agent Power Up. Each of the first three use an open set of protocols specified in the first message, while the last two use two fixed protocols.

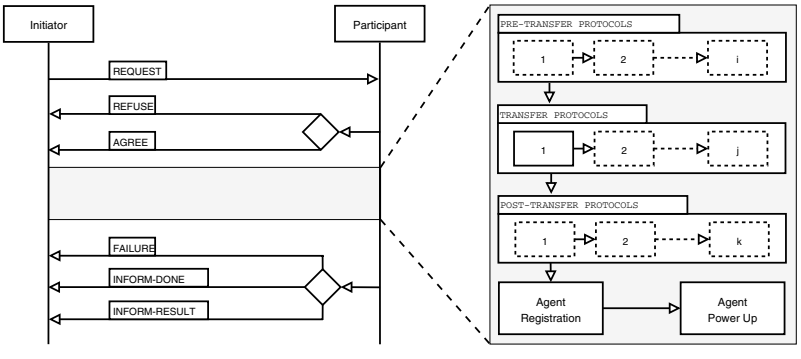


Fig. 1. Whole migration process

Main protocol. It starts and manages the whole migration process, including the rest of protocols following an IEEE-FIPA Request Interaction Protocol.The first message includes, among others, the ordered set of protocols desired in each one of the three first steps, as well as other agent requirements; the request may be refused or agreed.

Pre-Transfer, Transfer, and Post-Transfer steps. The functionality of these steps is the transfer of the agent code, data, and/or state plus optional operations to be done before and after the transfer. Although the functionality is clear, it can be implemented by different protocols chosen at runtime by the agent. As an example to illustrate a complete migration system, a transfer protocol called Agent Transfer protocol is provided.

Agent Registration and Agent Power Up steps. In the Agent Registration and Agent Power Up steps the agent is registered and restarted, respectively, in the destination platform. The agent is deleted from the source platform immediately after its registration. Only the fixed Agent Registration Protocol and Agent Power-Up Protocol, respectively, are allowed in these two steps.

3 Conclusions and Future Work

Mobile agent systems require new open and interoperable mechanisms for migration. In order to face this problem, we have presented a new open architecture for the future standardisation of agent migration based on several IEEE-FIPA agent specifications.

This architecture splits the migration process into three steps (Pre-Transfer, Transfer, and Post-Transfer), with a flexible open set of protocols in each step, plus two more steps (Agent Registration and Agent Power Up), each one with a fixed protocol. Specific protocols for the first three steps, which revolves around the agent code, data and/or state transferring, are unspecified allowing a new and wide range of strategies.

To demonstrate the feasibility of our proposed architecture, we have successfully implemented and tested it on JADE (as an evolution of the former migration service [1]) and AgentScape platforms, proving it is valid for completely different agent platforms. These two platforms share the same migration architecture but are based on different technologies, thus agents can migrate but cannot run in both. Future work will be focused on the agent execution interoperability with different platform technologies.

Acknowledgments. We want to thank J.R. Velasco and I. Marsa (UAH) for their suggestions. This work has been partially funded by the Spanish national project TSI2006-03481, the Catalan project 2005-FI-00752 and the European Social Fund (ESF).

References

1. Ametller, J., Cucurull, J., Martí, R., Navarro, G., Robles, S.: Enabling mobile agents interoperability through fipa standards. In: Klusch, M., Rovatsos, M., Payne, T.R. (eds.) CIA 2006. LNCS (LNAI), vol. 4149, pp. 388–401. Springer, Heidelberg (2006)
2. Braun, P., Rossak, W.: Mobile Agents. Morgan Kaufmann, San Francisco (2005)
3. Cucurull, J., Martí, R., Robles, S., Borrell, J., Navarro, G.: Fipa-based interoperable agent mobility proposal. arXiv:0706.1860v1 (2007)
4. FIPA: Fipa agent management support for mobility specification (2000)
5. FIPA: Fipa acl message structure specification (2002)
6. FIPA: Fipa agent management specification (2004)
7. Gray, R.S., Cybenko, G., Kotz, D., Peterson, R.A., Rus, D.: D'agents: applications and performance of a mobile-agent system. Softw. Pract. Exper. 32(6), 543–573 (2002)
8. JADE: Java Agent DEvelopment Framework (2007), `http://jade.cselt.it`
9. Lange, D.B., Oshima, M.: Mobile agents with java: The aglet api. World Wide Web 1(3), 111–121 (1998)
10. Overeinder, B.J., Brazier, F.M.T.: Scalable middleware environment for agent-based internet applications. In: Dongarra, J.J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 675–679. Springer, Heidelberg (2006)
11. Pinsdorf, U., Roth, V.: Mobile Agent Interoperability Patterns and Practice. In: Proceedings of Ninth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 238–244. IEEE Computer Society Press, Los Alamitos (2002)
12. Roth, V.: On the robustness of some cryptographic protocols for mobile agent protection. In: Picco, G.P. (ed.) MA 2001. LNCS, vol. 2240, Springer, Heidelberg (2001)
13. Vieira-Marques, P., Robles, S., Cucurull, J., Cruz-Correia, R., Navarro, G., Martí, R.: Secure integration of distributed medical data using mobile agents. IEEE Intelligent Systems 21(6) (2006)

HeCaSe2: A Multi-agent Ontology-Driven Guideline Enactment Engine

David Isern, David Sánchez, and Antonio Moreno

iTAKA Research Group - Intelligent Tech. for Advanced Knowledge Acquisition
Dept. of Computer Science and Mathematics. University Rovira i Virgili
43007 Tarragona, Catalonia (Spain)
{david.isern,david.sanchez,antonio.moreno}@urv.cat

Abstract. HeCaSe2 is a multi-agent system that intends to help doctors to apply clinical guidelines to their patients in a semi-automatic fashion. HeCaSe2 agents need a lot of (scattered) information on health care organisations, as well as medical knowledge, in order to provide an efficient support to health care practitioners. Modelling all these data is certainly a hard task. The paper describes how the inclusion of an especially designed ontology allows different agents to coordinate their activities in the enactment of clinical guidelines.

1 Introduction

A *clinical guideline* (GL) indicates the protocol to be followed when a patient is diagnosed a certain illness (*e.g.* which medical tests have to be performed on the patient to get further data, or what steps have to be taken according to the results of those tests) [1]. However, the inclusion of a guideline execution engine in the daily work flow of practitioners is a hard task. Taking this situation into consideration, in previous papers ([2]) we introduced a system called HeCaSe2 that proposes an open agent-based architecture, which represents different entities of a healthcare organisation. The system helps doctors to collect and manage information about patients and coordinates some complex tasks, such as scheduling meetings or looking for a required service. HeCaSe2 includes the management of clinical guidelines by doctors in the diagnosis or treatment of diseases. All these tasks require an external element to be more flexible and efficient: a representation of the care flow and the terminology used among all entities.

This paper presents a case of study that shows how practitioners are able to enact GLs through a distributed platform with an ontology that stores the medical knowledge. The designed ontology ([3]) has three main aims: *a*) to model healthcare entities with their relationships, *b*) to represent all medical terminology used by all partners, and *c*) to assign semantic categories to those medical concepts. With that approach, care is improved at least in three ways: *i*) *ontologies* provide a common understandable semantic framework to execute clinical guidelines; *ii*) *agents* can understand what they must perform at any moment

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 322–324, 2007.

and negotiate or coordinate their activities with the appropriate partners; and *iii*) ontologies provide a high level abstraction model of the daily work flow; that model can be *adapted* to each particular organisation, without the agents having to change their internal behaviour.

2 Ontology-Driven Clinical Guideline Execution

The combination of the designed medical ontology with the multi-agent system provides a flexible framework to follow the execution of clinical guidelines. In that process, two main tasks are required: *a*) to know the source of a data contained in an enquiry, and *b*) to identify the actor that provides an action and its result. In order to illustrate the procedure followed by agents, a simplified GL of *Deep Venous Thrombosis* (DVT) is used as an example. The GL was adapted from the National Guideline Clearinghouse and coded in *PROforma* [4]. The following case of study is focused in the diagnosis phase.

First of all, the doctor selects the DVT guideline from the repository (the descriptions of the available guidelines are provided by the Guideline Agent (GA)). The diagnosis is divided in two steps: 1) the doctor evaluates the risk to suffer a thrombosis-like pathology with different parameters, and 2) if the risk is moderate or high, the doctor makes some further tests. The associated Doctor Agent (DRA) analyses the GL and observes a first enquiry that includes nine parameters required in the risk evaluation. For each one, the DRA asks the Ontology Agent (OA) in order to obtain more details about it. The OA replies with the associated information contained in the Medical Ontology. Moreover, the DRA collects all available data in the patient health record (PHR) through the Medical Record Agent (MRA).

At this point, the DRA starts the execution of the GL by showing all the available data to the doctor, who can check all values and confirm any request to be performed. Imagine that the first required item is if the patient has any *Pitting Edema*. If the DRA does not find any information in the PHR about that concept, the DRA will make a request to the OA in order to know more details about it. OA responds that *Pitting_Edema* has the code *C0333243*, it has a Boolean result with a *false* default value, it is a *Sign_or_Symptom* (semantic type provided by UMLS), and it is responsibility of a *Family_doctor* or *Nurse* or *Physiotherapist*. With this data, the DRA determines that the doctor has to provide that Boolean result through the interface after asking the patient. Other concepts required to evaluate the risk, such as *Sore_to_touch*, *Swollen_calf*, *Swollen_legs*, *Superficial_vein*, *Immobile*, *Bed-ridden*, *Active_cancer* or *Changed_status*, are collected in the same way (by an examination of the patient or by collecting stored past values). All these values will remain in the PHR and they could be used in another visit or in other guidelines. Once all parameters have been filled, the guideline can continue the execution without the intervention of the doctor. The next task in the GL is a decision that evaluates the risk according to the values filled in the previous enquiry. The influence of each collected parameter allows the DRA to decide the risk to suffer DVT. At the end of that sub plan,

an internal variable stores that risk (low, medium or high). If the risk is medium or high, the doctor recommends a first test with *ultrasounds*.

The DRA requests the OA in order to know more details about the action *Ultrasound*. The OA responds that it is a synonym of the concept *Ultrasonography*, which has the code number *C0041618* which is a *Diagnostic_Procedure* that is responsibility of a *Radiologist_Service*. In that case, the DRA cannot accomplish that action because it requires another agent. The DRA begins a search through the same medical centre or outside. The DRA will eventually receive the name of the desired Service Agent (SA) (an instance of a *Radiology_service*) and they (the DRA and the SA) will negotiate a meeting for the patient to perform the visit. After that appointment, the doctor suspends the execution of the medical guideline until the action is performed and the results have been received. According to the results of that first test, the doctor can arrange an appointment to perform a venogram or not. In any case, as the result of that sub plan, an internal variable of the guideline will save the diagnostic result put by the doctor: *DVT Confirmed*, or *DVT Ruled out*.

3 Conclusions

A practical example of an ontology-driven guideline execution has been discussed. The use of ontologies in the medical domain is increasing and offers some advantages such as making domain assumptions explicit, separating domain knowledge from operational knowledge, and sharing a consistent understanding of what information means.

The presented work brings the following advantages to a guideline-based execution system: *a*) To identify the required actors that are able to accomplish an action and to know the source of a data, *b*) to adapt the execution framework to the particular structure of any healthcare organisation without modifying the MAS implementation, and *c*) to provide an application independent context. Thus, by changing the ontology and its relations, the execution procedure also changes. Note that the only issue that should be addressed is the manual definition of the appropriate task ontology. This question usually requires the intervention of a domain expert, but UMLS provides a large corpus of concepts and relations that can be easily reused.

References

1. Quaglini, S., Stefanelli, M., Cavallini, A., Micieli, G., Fassino, C., Mossa, C.: Guideline-based careflow systems. Artif. Intell. Med. 20, 5–22 (2000)
2. Moreno, A., Valls, A., Isern, D., Sánchez, D.: Applying Agent Technology to Healthcare: The GruSMA Experience. IEEE Intelligent Systems 21, 63–67 (2006)
3. Isern, D., Sánchez, D., Moreno, A.: An ontology-driven agent-based clinical guideline execution engine. In: Bellazzi, R., Ameen, A.-H., Hunter, J. (eds.) AIME 2007. LNCS (LNAI), vol. 4594, pp. 49–53. Springer, Heidelberg (2007)
4. Sutton, D., Fox, J.: The syntax and semantics of the PROforma guideline modeling language. J. Am. Med. Inf. Ass. 10(5), 433–443 (2003)

jTRASTO: A Development Toolkit for Real-Time Multi-Agent Systems

Mart Navarro, Vicente Julian, and Vicente Botti

Departamento de sistemas informticos y computacin
Universidad Politcnica de Valencia
Camino de Vera S/N 46022 Valencia (Spain)
{mnavarro,vinglada,vbotti}@dsic.upv.es

Abstract. Nowadays, there exist some toolkits that design and implement multi-agent systems. However, these toolkits have certain deficiencies when applied in specific environments (i.e. real-time environments). In these environment there may be strict temporal restrictions that require the use of specific agent architectures. Since these architectures are not covered in the development process of existing tools, this paper presents a new toolkit for the implementation of real-time multi-agent systems based on the jART platform.

1 Real-Time Multi-Agent System Development

The aim of this work is to provide a toolkit to support real-time, multi-agent system developments making the implementation process easier; and more dynamic and automatic so that the entire process is enhanced. This new toolkit, called jTRASTO (java Real-Time Agent Specification Toolkit) provides a powerful framework to develop real-time, multi-agent systems according to the jART (java Agent for Real-Time) platform [1]. The jTRASTO has been developed as an Eclipse plug-in [2]. This toolkit also supplies the necessary services to cover the deployment phase of the generated code.

Nowadays, most of the current developers of multi-agent systems use the Java programming language for agent implementation. However, Java is a totally inadequate language for use in real-time systems [3]. Some of the characteristics that make Java inappropriate are the garbage collector and dynamic load of the classes, which introduce unpredictability to the system, and priority inversion problem in resource sharing. An extension of the Java language, called Real-Time Java, emerged to resolve these problems in real-time condition. The RTSJ approach [4] is the most well-known related work and offers a specification with the needed extensions. The most important improvements of the RTSJ are: thread scheduling, memory management, synchronization, asynchronous events, asynchronous flow of control, thread termination, and physical memory access.

Therefore, we have developed the jART platform using the RTSJ specification. The platform allows the inclusion of temporal constraints for different services offered by agents inside the platform. These agents with temporal constraints

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 325–327, 2007.

(real-time agents) interact with the rest of agents in a real-time environment. The platform provides the mechanisms necessary for the creation, control and communication of agents. All the services, components and auxiliary agent in the platform are already implemented as real-time tasks whose executions are predictable and temporally constrained.

The creation of this platform follows the specification defined by the FIPA organization [5]. Figure 1 shows the connection of the distinct modules needed to execute the jART platform. For reasons of brevity, a description of the platform and its components can be consulted in more detail in [1].

2 A Toolkit to Develop jART Agents

The implementation of systems using the jART platform requires knowledge about real-time systems or real-time java libraries. In order to facilitate the use of this platform we have developed a toolkit. This new toolkit, called jTRASTO, allows the implementation of real-time multi-agent systems to be executed on the jART platform. This toolkit allows the creation of real-time multi-agent systems without the developer having to learn a new language, such as Real-time java. It also hides the creation and control agent methods, the handling of real-time thread associated to tasks, memory management, and access to shared resources.

Figure 2 shows the different steps required to create a real-time multi-agent system using the jTRASTO toolkit. Briefly, once the analysis phase (step A) and design phase (step B) of the system have been completed. The designed will create the agents that will participate in the real-time multi-agent system. Once all the agents have been created (step C), the designer will define the different behaviours that each agent can carry out (step D) and will define the interactions between the agents (Step E). This plug-in allows the designer to define the communication protocols that the agents will follow during their interaction (step E). Each communication protocol will be initiated by a behaviour of an agent. Once the agents, their behaviours and the different interactions among them are defined, a first version of the real-time multi-agent system can be completed and executed. There are two options for starting up the real-time

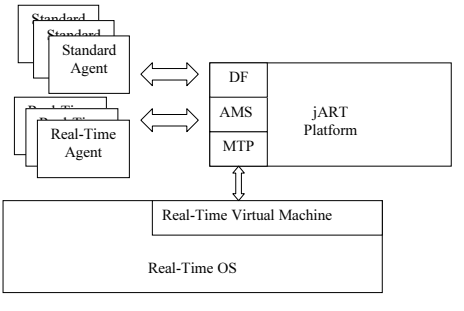


Fig. 1. Main System architecture

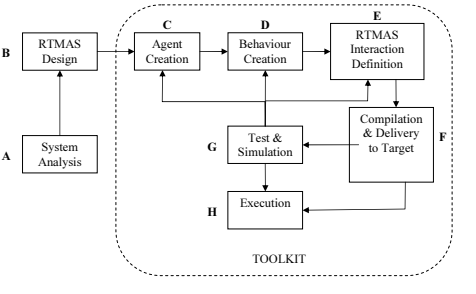


Fig. 2. Development steps of the jTRASTO toolkit

multi-agent system (Step F). The first option is to generate a java bytecode and execute it on a real-time virtual machine that is running on a RTOS. The second option is to convert the java code into native code using a GCJ compiler [6]. This native code can be executed on a POSIX [7] RTOS.

The jTRASTO toolkit carries out behaviours traces during execution (Step G). These behaviour traces of the agents within the system can be visualized using the Kiwi toolkit [8]. These trace provide information about the activation and completion of the behaviours, their maximum execution frequency, and their deadlines. In addition, the traces indicate whether or not their deadlines have been fulfilled. Next, this first version of the real-time multi-agent system can be executed on the target system (Step H).

3 Conclusions

This paper has presented a toolkit to develop real-time, multi-agent system using the jART platform. With this toolkit, developers can create agents and theirs associated behaviours as well as defining the interactions among them. Developers can choose between two types of compilation: either execution on a real-time virtual machine, or directly without a virtual machine over RTOS.

We are currently working on an improved version of the jART platform to include argumentation-based negotiation with real-time constraints inside the real-time, multi-agent system.

Acknowledgements

Support received from the Spanish government and FEDER funds under CICYT TIN2005-03395 and TIN2006-14630-C0301 projects is gratefully acknowledged.

References

1. Navarro, M., Julián, V., Soler, J., Botti, V.: jART: A Real-Time Multi-Agent Platform with RT-Java. In: IWPAAMS'04, pp. 73–82 (2004)
2. Eclipse Platform Technical Overview (2003), `http://www.eclipse.org/whitepapers/eclipse-overview.pdf`
3. Wellings, A.: Concurrent and Real-Time Programming in Java. Wiley, Chichester (2004)
4. The Real-Time for Java Expert Group, `http://www.rtj.org`
5. FIPA Abstract Architecture Specification, `http://www.fipa.org/specs/fipa00001`
6. GCJ web, `http://gcc.gnu.org/java/`
7. González, M.: Real-Time POSIX: An Overview. VVCONEX-93 (1993)
8. Kiwi web page, `http://rtportal.upv.es/apps/kiwi/`
9. Timesys web, `http://www.timesys.com`

Models and Tools for Mulan Applications

Lawrence Cabac, Till Dörges, Michael Duvigneau,
Christine Reese, and Matthias Wester-Ebbinghaus

University of Hamburg, Department of Computer Science,
Vogt-Kölln-Str. 30, D-22527 Hamburg
http://www.informatik.uni-hamburg.de/TGI

Abstract. In this work we describe the development process of multi-agent application design and implementation with Mulan. Our approach can be characterized as model driven development by using models in all stages and levels of abstraction regarding design, implementation and documentation. Both, standard methods from software development as well as customized ones are used to satisfy the needs of multi-agent system development.

1 Introduction

The agent metaphor is highly abstract and it is used to develop software engineering techniques and methodologies that particularly fit the agent-oriented paradigm. Flexibility and autonomy of an agent's problem-solving capabilities, the richness of agent interactions and the (social) organizational structure of a multi-agent system as a whole must be captured.

Our approach borrows several ideas from well-known methodologies (AO, OO) as well as concepts from conventional modeling techniques (UML). We integrate our techniques and tools with the model-based implementation of the Mulan framework, which facilitates Petri net-based programming. In the Mulan architecture, agents and multi-agent systems are modeled through the high-level Petri net formalism of reference nets.

The result of those efforts is a development methodology that continuously integrates our philosophy of Petri net-based and model-driven software engineering in the context of multi-agent systems. In Section 2 we introduce the basic conceptual features of multi-agent application development with Mulan. The particular techniques, models and tools are presented in Section 3.

2 Concepts of Application Development with Mulan

The reference net-based multi-agent system architecture Mulan (Multi Agent Nets) structures a multi-agent system in four layers, namely *infrastructure*, *platform*, *agent* and *protocol* [2,3]. In a multi-agent application the organizational structure has to be defined, such that responsibilities for all aspects of the system are specified. The general perspectives in the area of multi-agent systems are the

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 328–330, 2007.

structure, the interactions, and the terminology. These perspectives are orthogonal with connecting points at some intersections. The structure of a multi-agent system is given by the agents, their roles, knowledge bases and decision components. The behavior of a multi-agent system is given by the interactions of the agents, their communicative acts and the internal actions related to the interactions. The terminology of a multi-agent system is given as a domain-specific ontology definition that enables agents and interactions to refer to the same objects, actions and facts. Without a common ontology, interactions are impossible.

3 Techniques, Models and Development Tools

In this section we describe the techniques applied during the various stages of multi-agent application development with Mulan. Most of the techniques and tools mentioned can be seen in Figure 1.

The *coarse design* is done mainly in open discussions. The results are captured in simple lists of system components and agent interactions. This culminates in a *use case diagram*. The *structure of the multi-agent application* is refined using a *role diagram*. This kind of diagram uses features from class diagrams and component diagrams. The terminology of a multi-agent system is used in form of an *ontology definition* by the agents to communicate with each other and

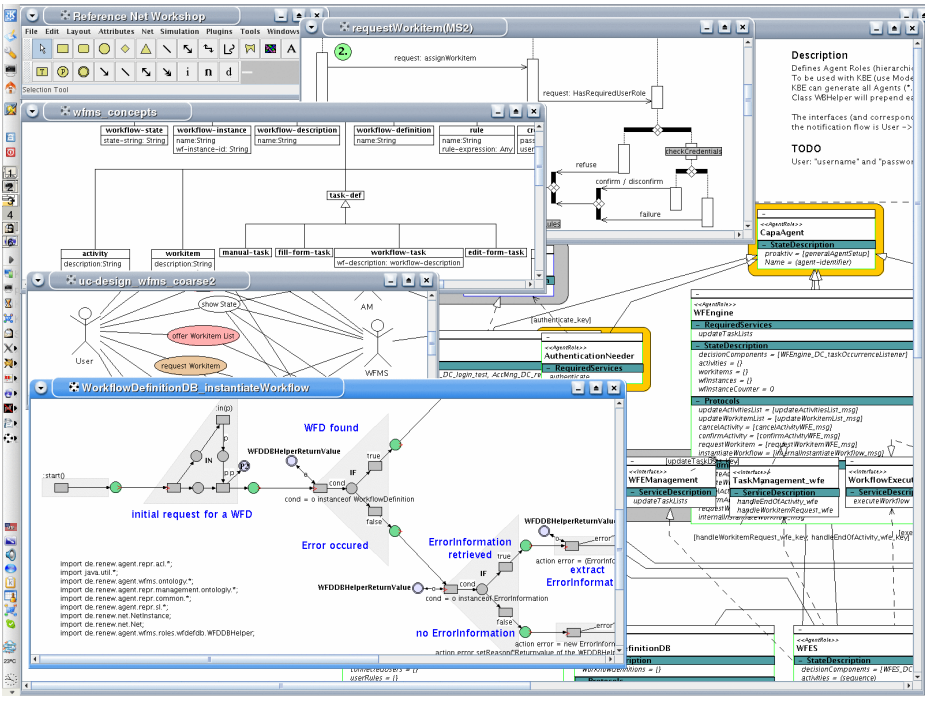


Fig. 1. Models

for their internal representation of the environment. The facts about an agent's environment are located in its *knowledge base*. The *behavior* of the system components is specified using *agent interaction protocol diagrams* (AIP, integrated into PAOSE in [1]). Additionally *Decision Components* (DC) can be used to encode behavior not directly related to communication between agents. Last, but not least *monitoring and debugging* tools are developed in our group.

4 Conclusion

In this paper we presented an integrated approach to multi-agent application development with Mulan, which is based on Petri nets and agent-oriented software engineering. The approach allows for multiple levels of abstraction. The tools that are used during the development process support all phases of development with modeling power and deployment facilities, although some of the tools still have prototypical character.

For the future, we follow several directions to refine the approach. On the practical side, we look into further developments, improvements and integration of tools and techniques. On the conceptual side, we want to expand the multi-agent oriented approach to other aspects of the development process like project organization and agent-oriented tool support. Following these directions, we want to achieve symmetrical structures in all three aspects of software development: the system, the development process and the project organization.

References

1. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring Petri net-based agent interaction protocols. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 102–120. Springer, Heidelberg (2003)
2. Köhler, M., Moldt, D., Rölke, H.: Modelling the structure and behaviour of Petri net agents. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 224–241. Springer, Heidelberg (2001)
3. Rölke, H.: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen. Agent Technology – Theory and Applications, vol. 2. Logos Verlag, Berlin (2004)

Multi-agent Architecture for Intelligent Tutoring Systems Interoperability in Health Education *

Carolina González [1,2], Juan C. Burguillo[1], and Martín Llamas[1]

[1] Departamento de Telemática, Universidad de Vigo, Spain
[2] Departamento de Sistemas, Universidad del Cauca, Colombia
{cgonzals,jrial,martin}@det.uvigo.es

Abstract. Intelligent Tutoring Systems (ITSs) base their interoperability on the interchange of domain knowledge about learning and teaching processes and about students. To enable the interchange of domain knowledge between ITSs and heterogeneous Health Information Systems (HIS) we propose a multi-agent architecture allowing sharing patient's clinical data in Health Information Systems. The Health Level (HL7) is used as a standard messaging protocol for collecting the clinical patient data distributed over Health Information Systems.

Keywords: Intelligent Tutoring Systems, Multi-agent systems, HL7, Health Education, interoperability.

1 Introduction

Intelligent Tutoring Systems (ITSs) has been developed also in the domain of health education. The purpose of these systems is to provide rich environments for maximizing learning while minimizing risks to patients, until sufficient competency is established. In order to provide good quality in health education, the International Medical Informatics Association (IMIA) has proposed a set of recommendations. One of the key principles of IMIA, is the need to provide the required theoretical knowledge, practical skills and mature attitudes. In this sense, we consider the importance of integrating Health Information Systems (HIS) and ITSs for providing sensible feedback and explanations to the student. Feeding the ITSs knowledge base with real cases, the student in addition to provide theoretical knowledge, would acquire the practical skills to interact with real cases reaching the necessary abilities for improving the decision-making process. In this paper, we propose a multi-agent architecture for sharing patient's clinical data among ITSs and heterogeneous HIS.

2 Multi-agent Architecture for ITS Interoperability

In a previous work we have proposed a framework for designing ITSs in Health Care Domain, using Case-based reasoning (CBR) methodology and Multi-Agent Systems

* The work was funded by the project MetaLearn (TIN2004-08367-C02-01).

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 331 – 333, 2007.

(MAS) [1]. In this new approach an integration component for supporting ITSs interoperability with HIS is proposed. Fig. 1 shows each one of the components of the proposed framework and details the HL7 Message Server as the new integration component. In order to integrate ITSs and HIS we have developed a variety of functions and we have adopted the HL7 standard [2] to create a health care messaging and exchange data among the systems. In the internal structure of the system each entity of HL7 is implemented as an agent, which is delegated a specific function and each agent is on a multi-agent platform. In this platform, individual and heterogeneous systems can communicate and share the patient's information. The HL7 Message Server is composed by:

a. Event manager agent: it permits the message parsing and construction, and also sends control management message to the system. The most important functions are as follows: (a) message queue, (b) message mapping, (c) message parsing; (d) message building, and (e) message control. The agent receives an HL7 event notification encoded in XML from the messages interface and according to the event type; it classifies the incoming events into an appropriate event queue.

b. HL7 transformer agent: it assembles automatically HL7 messages. Upon notification of a new HL7 trigger event, the HL7 transformer requests the data loader to fetch relevant data from message view. It automatically generates appropriate HL7 messages using predefined mapping functions, and it communicates with the different Health Information Systems.

c. Message sender: it transfers HL7 messages to the external systems; it receives acknowledgments messages for verification purposes. These acknowledgments messages guarantee a successful communication of the messages.

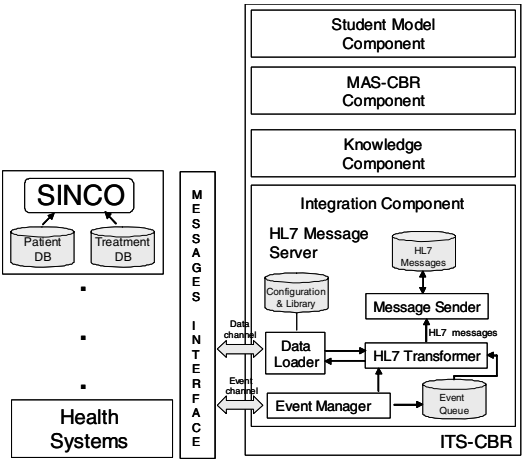


Fig. 1. Multi-agent Architecture for ITS interoperability

In our agent based approach the information in each message is embedded in the agent properties. Depending on the HL7 trigger event that initiated the creation of the agents, the HL7 agents contain all the information that would have been part of a corresponding HL7 message. In the agent communication process, the HL7 messages

are encoded in the content tag with XML. In the first phase of HL7, the trigger events are defined using use cases. The secondary phase produces means of specifying responsibilities of the senders and receivers. The message specification is based on the domains of Patient Administration and Care Provision, defined according to the HL7 specification.

Finally, the aforementioned architecture was validated in a scenario integrating a Health Information System for prevention and control of tuberculosis (SINCO-TB) and an Intelligent Tutoring System for Health Education (STIM-Tutor) [3].

3 Discussion and Conclusions

Our main contribution in this paper is the description of a multi-agent architecture to share patients' information among ITSs and HIS using HL7. The knowledge is acquired from real patient cases improving the medical students' learning process. The agents' technology allows obtaining the information about patients in HIS. Our agents cooperate to solve mappings between local database applications and the HL7 message templates; this cooperation improves the system functionality. The agent communication language provides a standard syntax and semantics for information exchange between ITS and HIS components. This simplifies the exchange of patients' information and the training of new students in complex and real situations using ITSs.

The use of an established health care standard, HL7, to share domain information among systems, is particularly significant for healthcare organizations that were assessed towards its adoption. The incorporation of XML-based HL7 messages in the agent architecture, provides the mechanism for disseminating patient information based on the HL7 event model in an effectively way. With the use of HL7 it is possible to have access to the information of each Health Information System only using the HL7 interface, i.e. hiding and encapsulating the internal structure of the systems' database. Future work concerns the evaluation of this approach from the learner view to enhance the learner performance.

References

1. González, C., Burguillo, J.C., Llamas, M.: A case-based approach for building Intelligent Tutoring System. In: 7th International Conference on Information Technology Based Higher Education and Training, Sydney, Australia, pp. 1–5. IEEE, Los Alamitos (2006)
2. HL7: What is HL7 (2002), http://www.hl7.org/about/
3. González, C., Burguillo, J.C., Llamas, M.: Integrating Intelligent Tutoring Systems and Health Information Systems. In: 1st International Workshop on E-learning Systems, Regensburg, Germany, IEEE Computer Society Press, Los Alamitos (2007)

Multi-agent Planning in Sokoban

Matthew S. Berger and James H. Lawton

US Air Force Research Laboratory, Information Directorate, Rome, NY, USA
{Matthew.Berger,James.Lawton}@rl.af.mil

Abstract. The issue of multi-agent planning in highly dynamic environments is a major impediment to conventional planning solutions. Plan repair and replanning solutions alike have difficulty adapting to frequently changing environment states. To adequately handle such situations, this paper instead focuses on preserving individual agent plans through multi-agent coordination techniques. We describe a reactive agent system architecture in which the main focus of an agent is to be able to achieve its subgoals without interfering with any other agent. The system is a 3-level architecture, where each level is guided by the following fundamental principles, respectively: *when* is it valid to generate a plan for a subgoal, *who* is most appropriate for completing the subgoal, and *how* should the plan be carried out.

1 Introduction

Multi-agent planning in the context of rapidly-changing environments can quickly become an involved, unmanageable problem. Further, maximizing concurrency and ensuring conflict-free agent plans becomes a difficult problem to manage when using techniques like plan repair and replanning. Thus, multi-agent coordination through conflict resolution of individual agent plans can be an effective way to ensure safe planning while providing a reasonable execution time.

This paper describes an approach that rapidly merges individual agent plans in highly dynamic environments. Agents generate their own plans separately, and use a coordination process that is concerned with maximizing parallelism and minimizing conflicts in the execution of the plans. The algorithm is targeted towards environments in which a global goal may be difficult for a single agent to achieve alone, but becomes more manageable when the goal can be partitioned among several agents. When each generates a plan for its own subgoals, the problem reduces to agent coordination, in ensuring that no conflicts exist between plans.

For environments in which resources have high levels of contention, it is unlikely that each agent's plan can be executed without conflicting with some other agent's plan. Thus, we propose a system architecture capable of dealing with such environments, consisting of two basic principles: *goal abstraction* and *flexible agent behavior*. Goal abstraction refers to decomposing a plan into a hierarchy of subgoals to be achieved by the plan. In contentious environments, coordinating separate agent subgoals is far more effective than relying on primitive actions, as enforcing synchronization actions on subgoals ensures that plans

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 334–336, 2007.

expanded on these subgoals will be conflict free [1,2]. Flexible agent behavior refers to agents not being bound to their initial plans. An agent bound to its plan in a contentious environment will find it difficult to replan and/or use plan-repair as time progresses [3]. The combination of goal abstraction and flexible agent behavior is well suited for highly dynamic environments: as subgoal conflicts are worked out, agents are aware of subgoals to be achieved, even if they initially belong to another agent, and plan from there. Note the distinction between subgoals and partial plans, where partial plans commonly characterize hierarchical planning. It is implicit that when a partial plan is refined, it still belongs to the agent responsible for it. An agent may own a subgoal, but is not necessarily bound to the plan that results in the subgoal.

Motivated by these observations we have developed a 3-level multi-agent coordination architecture (described in Section 2). Each level resolves fundamental issues in agent conflicts. The first level addresses *when* it is valid to plan for a subgoal by considering temporal consistency of the subgoals. The second level handles the issue of *who* should be made responsible for achieving a subgoal, in our case using an auction mechanism. The third level considers *how* to achieve a subgoal, in which agents reserve resources for planning out primitive actions.

1.1 Multi-agent Sokoban

To ground this discussion we use the domain of Sokoban as a running example to better illustrate the system. Sokoban is a simple discretized box pushing environment in which movers must push all boxes in the environment to goal squares. Despite its simple premise, the game becomes extremely difficult to manage as the number of boxes and the size of the environment increase. It has in fact been shown that the game is PSPACE-Complete [4].

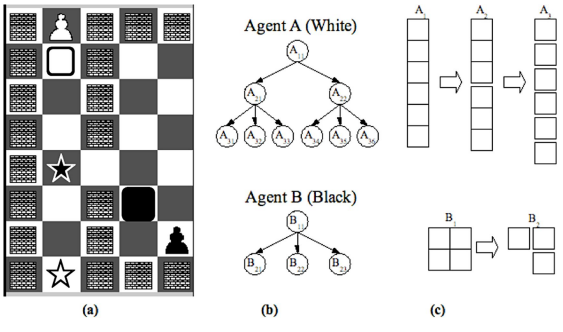


Fig. 1. (a) An example Sokoban scenario. The white agent must push the white box all the way down, while the black agent must push the box up, then push it to the left. (b) The resulting subgoal hierarchies for both agents. (c) Resource usage for all subgoals.

Sokoban in the context of a single agent becomes an unmanageable problem rather quickly, therefore motivating the desire to split the problem up among multiple agents. If we assign a subset of boxes to each agent, in which each agent is responsible for producing a plan that solves its separate goals, the problem now becomes more manageable as we are significantly limiting the search space for each agent. Our main objective now becomes ensuring that each agent may execute its plan conflict free with regard to the other agents.

2 Coordination Architecture

Determining *when* is it valid to plan for a subgoal in our architecture is a matter of verifying that subgoals are *temporally consistent* with respect to one another. Assume each agent A_i has generated a plan P_i to achieve its goal S_i, where their are n agents and the global goal G is decomposed into separate goals for each agent, such that $G = S_1 \wedge S_2 \wedge ... \wedge S_n$ (see Figure 1b). We can then use *temporal constraint matrices*, which identify when agents intend to use various resources, to identify both conflicts and potential deadlock conditions in the resource requirements of the various plans P_i.

Once each agent has found some subgoals $Q_i \subset S_i$ to be temporally consistent, the problem reduces to *which agent* should be responsible for producing a plan for each subgoal. We use an auctioning approach in which bids – computed using current work loads and available resources – are solicited from other agents to determine which agent can best satisfy a given subgoal. As subgoals are auctioned off they are removed from the agent's agenda. Subgoals where no agent places a valid bid are reconsidered in the next auction cycle.

Finally, we handle *how* a subgoal can be guaranteed to be conflict free via *resource reservations*. For a given subgoal $q \in Q_i$, each agent A_i broadcasts information about the resource needs of q, and uses this information to compute resource conflicts. An agent may then plan for a subgoal following: (i) it may not use resources requested by another agent with a higher priority, and (ii) the cost of the plan must not exceed the smallest lower bound on the given resources. In the multi-agent Sokoban domain, we identify "zones" of spaces in an agent's planned path as the resources of the plan (Figure 1c).

3 Conclusion

We have presented an architecture for multi-agent planning in highly dynamic environments. In the future we plan to address *subgoal alternatives* that incorporate *or* plans into the current architecture, as in [1], as well as to resolve conflicts that require replanning or plan-repair. We believe that temporal constraint matrices may be used to efficiently guide plan-repair in these situations.

References

1. Clement, B., Durfee: Top-down search for coordinating the hierarchical plans of multiple agents. In: Proc. of the Third Annual Conference on Autonomous Agents, pp. 252–259 (1999)
2. Sugawara, T., Kurihara, S., Hirotsu, T., Fukuda, K., Takada, T.: Predicting possible conflicts in hierarchical planning for multiagent systems. In: Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 813–820 (2005)
3. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan stability: Replanning versus plan repair. In: Proceedings of International Conference on AI Planning and Scheduling (ICAPS-06) (2006)
4. Culberson, J.: Sokoban is pspace-complete. Technical Report TR97-02, University of Alberta (1996)

Ontology Matching in Communication and Web Services Composition for Agent Community

Adam Łuszpaj, Edward Nawarecki, Anna Zygmunt, Jarosław Koźlak, and Grzegorz Dobrowolski

Department of Computer Science, AGH-UST,
Al. Mickiewicza 30, 30-059 Kraków, Poland
{luszpaj,nawar,azygmunt,kozlak,grzela}@agh.edu.pl

Abstract. Ontology matching plays a vital role in a number of areas, such as knowledge integration, services composition or system interoperability. In this paper we present an architectural overview of an ontological service, wrapped up as an Ontology Agent, supporting automated ontology matching and facilitating inter-agent communication.

1 Introduction

Ontology matching plays a vital role in a number of areas, such as knowledge integration, services composition or system interoperability. In this paper we present an architectural view of an ontological service supporting automated ontology matching and facilitating inter-agent communication. The aims of this service, provided by the so-called Ontology Agent (referred to as OA), in compliance with the FIPA specification, are: acquiring public ontologies, translating expressions between ontologies, providing information regarding relations between concepts and between ontologies, mediating between agents in order to establish a common ontology. The effectiveness of such a service depends on the underlying algorithms for discovering semantic correspondences between ontology concepts. Such a service can also be useful for discovery and composition of semantically annotated web services. Section 2 presents an overview of a general architecture of the OA. In section 3 we briefly sketch a concept of Web Services Composition Agent (referred to as WSCA) providing a service for composing atomic web services described in compliance with the OWL-S standard. Section 4 contains a brief summary of our research.

2 Ontological Service Architecture

In this section we enumerate the main building blocks for the OA, whose architecture is presented in Fig. 1.

Ontologies Acquisition Module. This part of the OA is responsible for the acquisition of ontologies available on the Semantic Web. The task can be delegated to a specialized agent identifying OWL/RDFS documents and sending back the results, which

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 337 – 339, 2007.

are in turn input into the registration module. Acquisition can take place in several ways: automatic acquisition of the document published on the Internet or downloading ontologies directly into the registration module from another system.

Registration Module. Acquired ontologies are stored in the ontological repository based on relational databases, which guarantees quicker access and better data integrity. The underlying matchmaking algorithms, which are outside the scope of this paper, are schema-driven. This is the reason for not acquiring an assertional part of ontologies. At the registration stage of a particular ontology, the registration module, supported by the matching module, determines mappings between ontologies and encodes them in the properly designed structure.

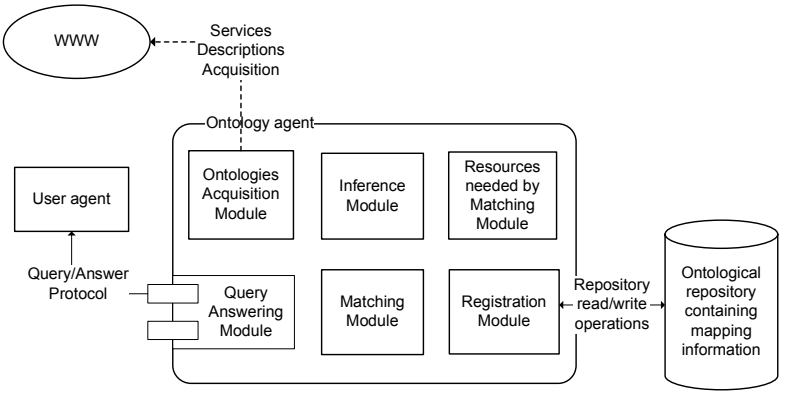


Fig. 1. The OA modules

Inference Module. The function of this module is to provide inference services for the OWL ontologies. Its functioning is based on a reasoner operating on a DL ontological model. TBox reasoning infers information about implicit relations between concepts, which is necessary both at the stage of concept mapping and at run-time, when processing queries.

Query Answering Module. This is an interface module. With its help, higher level services can make use of the knowledge encoded in ontologies and mappings between them. The module processes and answers queries concerning terminological knowledge and regarding mappings between concepts.

Mtching Module. This is the key module. It wraps algorithms for finding semantic correspondence between concepts and between ontologies. Identified mappings are then stored in the database. The correspondence between ontologies is characterized by the type defining the similarity level between a pair of ontologies, according to the FIPA specification of an ontological service.

Mappings between concepts deriving from particular ontologies are twofold:

- Equivalence (*owl:equivalentClass*)
- Subsumption (*rdfs:subClassOf*)

At the registration stage of a new ontology, there is one important issue: the choice of the ontology and concepts for which mappings are to be found. Analyzing all available ontologies would be very time-consuming and, in most cases, pointless, therefore an upper ontology is introduced. It plays a vital role in the decision process of mapping algorithms. Apart from identifying relations between domain ontologies' concepts, the module also matches the registered ontology to the upper ontology's structure, likewise using equivalence and subsumption mechanisms. As a result, the scope of potential concepts, which can be mapped, is limited to certain sets of concepts semantically located in the vicinity of the registered ontology.

3 Web Services Composition

OWL-S ontologies define the mature standard of semantic annotation helping in automated discovery, execution and composition. Defining service parameters in the context of ontologies helps to match the client's needs with offered services more accurately. However, the greatest advantage is the possibility of composing many atomic services into a complex functionality, adjusted to specific requirements. The composition service, wrapped by the so-called Web Services Composition Agent, makes use of the OA's matching capabilities in order to compose various services. IO parameters of services, originating from various ontologies and exploiting semantic mapping, constitute the basis for composition.

4 Conclusion

Agents and services with explicitly expressed semantics are expected to be one of the building blocks of what is called the Semantic Web. The main obstacles are the difficulties in interoperability resulting from a heterogeneity of conceptualizations, ontologies or applied languages. In this paper we have presented an agent infrastructure providing services for mappings discovery between ontologies in order to facilitate interoperability between heterogeneous and autonomous Web resources.

References

1. Bansal, S., Vidal, J.: Matchmaking of Web Services Based on the DAML-S Service Model. In: Sandholm, T., Yokoo, M. (eds.) AAMAS'03. Second International Joint Conference on Autonomous Agents, pp. 926–927. ACM Press, New York (2003)
2. van Diggelen, J., Beun, R.J., Dignum, F.P.M., van Eijk, R.M., Meyer, J-J.C.: Combining Normal Communication with Ontology Allignment. In: Dignum, F., van Eijk, R.M., Flores, R. (eds.) AC 2005. LNCS (LNAI), vol. 3859, pp. 17–32. Springer, Heidelberg (2006)
3. Shvaiko, P., Euzenat, J.: A Survey of Schema-based Matching Approaches. Journal on Data Semantics 4, 146–171 (2005)

Plugin-Agents as Conceptual Basis for Flexible Software Structures

Lawrence Cabac, Michael Duvigneau, Daniel Moldt, and Benjamin Schleinzer

University of Hamburg, Department of Computer Science,
Vogt-Kölln-Str. 30, D-22527 Hamburg
http://www.informatik.uni-hamburg.de/TGI/

Abstract. To allow for flexibility in software structures (architectures) especially plugins and agents are proposed solutions. While plugins are used to support the conceptual and practical issues within component oriented software environments, agents are used in software areas where social metaphors like (self-)adaptability, flexibility, mobility, interactivity etc. are of interest. Common to both approaches is a strong relation to a service-oriented view on exporting functionality. This contribution illustrates the idea of the integration of both concepts on the formal basis of high-level Petri nets.

Keywords: High-level Petri nets, Nets-within-nets, reference nets, Renew, plugins, Mulan, multi-agent systems, plugin-agents.

1 Introduction

While plugins and agents differ considerably in the current common perspective of software developers, there are some important similarities. On one side the area of component oriented computing usually concentrates on the flexible composition of software parts. Additional functionality is the main focus (see [1,2] for a thorough discussion of the concept). On the other side agents focus on the powerful modeling metaphors and concepts of autonomous behavior, mobility, pro-activeness, and communicative abilities [3]. There have been approaches to combine (mainly technical) aspects from both areas, plugin systems and agents technology [4,5]. This contribution now provides a more conceptual integration based on the formal model of high-level Petri nets (reference nets [6]). The integration can be considered as plugin-agents.

2 Conceptual and Technical Background

The three parts *Java*, Renew and Capa provide a runtime environment for plugin-agents and applications build on top. Renew can execute a multitude of net formalisms like reference nets concurrently and is written in *Java*. Reference nets combine object-based and nested modeling power with true concurrency. Capa is a Fipa compliant multi-agent-system that's implemented in reference nets and (pure) *Java*.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 340–342, 2007.

Reference nets [6] are based on the colored Petri net formalism. They extend the colored Petri net formalism by combining the concepts of synchronous channels and the nets-within-nets concept by Valk [7]. In addition, reference nets [8] allow for net instances, dynamically changing nested net structures and the ability to include *Java* inscriptions. Renew [9] is a tool for the construction and execution of reference nets and their inspection (monitoring / debugging) at runtime. Due to the nets-within-nets paradigm, thus allowing to model active tokens embedded in an environment, the nested structures and the ability to dynamically change that structure, reference nets are a good choice to model plugins and agents.

Agents are designed to handle problems of conflicting functionality, service dependencies, locality and privacy, compatibility and dynamic extensibility [10]. Agents have the ability to act independently and autonomously (i.e. they decide on their reactions to inputs and act pro-actively, see [3]). Based on [11] Mulan can be used to build highly abstract models based on agent concepts.

Plugins can be seen as dynamic components [12]. In the same way as components, plugins provide functionality extension to enhance the software system [2]. However, plugins can achieve this goal in a dynamical manner by the mechanism described through the *plugin* metaphor. All this is supported by a plugin management system (PMS), which provides an interface to register, deregister and publish services of the plugins in the system. Each service has a description of how other plugins can utilize that service. The PMS also handles dependencies between plugins (compare [13]).

We combine advantages of both areas: agents and plugins. Agents do not allow to extend the functionality of other agents (entities in the system). They can nevertheless add to the functionality of a platform. By adding platform functionality to an agent, it can be extended by a pluggable agent. This idea leads to a natural inclusion of the plugin concept into agent technology.

Several advantages arise from the concept of a plugin-agent. We can assure locality for an agent. Usually in a multi-agent system, it is not possible to ensure that an addressed agent is located at a given platform. Response times can be reduced by using direct (synchronous) communication instead of time consuming (asynchronous) message handling in interactive systems. Nowadays, hardly any user accepts GUI interfaces that are not responsive. Locality and direct (proprietary) communication can ensure a secure connection between two entities. In mobile environments (mobile multi-agent system) user-personalized plugins are able to provide sensitive personal data and personalized functionality to mobile users, which can also connect at distinct and versatile platforms to the multi-agent system.

3 Conclusion

The combination of agent technology and plugin technology seems tempting. Advantages of both worlds can be used together to achieve more flexibility while increasing clearness at the same time. Agent technology provides autonomous

behavior, pro-activity, distribution and even mobility, while plugin technology provides well established concepts for light-weight extensible architectures.

Through explicit modeling of the concepts (plugin, agent and plugin-agent) we achieve a clear and well founded architecture. The visual models enable us to easily transfer conceptual knowledge to other developers. The operational semantics of the Petri net models allows us to refine the models to obtain a full fletched running agent framework providing those concepts. The executable versions allow for application development (rapid prototyping).

References

1. Sametinger, J.: Software Engineering with Reusable Components. Springer, Heidelberg (1997)
2. Schumacher, J.: Eine Plugin-Architektur für Renew – Konzepte, Methoden, Umsetzung. Master's thesis, University of Hamburg (2003)
3. Wooldridge, M.: Intelligent agents. Multiagent systems: a modern approach to distributed artificial intelligence, pp. 27–77 (1999)
4. Minh Vu, C.T.: E2 agent plugin architecture. In: 2005 International Conference on Integration of Knowledge Intensive Multi-Agent Systems, KIMAS'05: Modeling, Exploration, and Engineering (2005)
5. Tu, M.T., Griffel, F., Merz, M., Lamersdorf, W.: A plug-in architecture providing dynamic negotiation capabilities for mobile agents. In: Rothermel, K., Hohl, F. (eds.) MA 1998. LNCS, vol. 1477, pp. 222–231. Springer, Heidelberg (1998)
6. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
7. Valk, R.: Petri nets as token objects - an introduction to elementary object nets. In: Desel, J., Silva, M. (eds.) ICATPN 1998. LNCS, vol. 1420, pp. 1–25. Springer, Heidelberg (1998)
8. Kummer, O.: Introduction to Petri nets and reference nets. Sozionik Aktuell 1, 1–9 (2001)
9. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – User Guide. University of Hamburg, Faculty of Informatics, Theoretical Foundations Group, Hamburg. Release 2.0 edn. (2004), available at `http://www.renew.de/`
10. Cabac, L., Duvigneau, M., Moldt, D., Rölke, H.: Agent technologies for plug-in system architecture design. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, Springer, Heidelberg (2006)
11. Rölke, H.: Modellierung von Agenten und Multiagentensystemen – Grundlagen und Anwendungen. Agent Technology – Theory and Applications, vol. 2. Logos Verlag, Berlin (2004)
12. Eichler, C.: Entwicklung einer Plugin-Architektur für dynamische Komponenten. Master's thesis, University of Hamburg (2002)
13. Cabac, L., Duvigneau, M., Moldt, D., Rölke, H.: Modeling dynamic architectures using nets-within-nets. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 148–167. Springer, Heidelberg (2005)

Selection of Efficient Production Management Strategies Using the Multi-agent Approach

Jarosław Koźlak[1], Jan Marszałek[1,2], Leszek Siwik[1], and Maciej Zygmunt[2]

[1] Dept. of Computer Science, AGH-UST, Al. Mickiewicza 30, 30-059 Kraków, Poland
[2] ABB Corporate Research Center, ul. Starowislna 13A, 31-038, Kraków, Poland

Abstract. This work focuses on the problem of inventory management which is an important element of the supply-chain management challenge. The goal of our research was to develop an application which supports decision making regarding optimal strategies for producing electrical machines. The multi-agent approach offers several valuable features which may be exploited for supply-chains management: problem decentralization, isolation of different kinds of sub–problems and solving them separately, as well as modeling and predicting the behavior of the particular modules and referring to the means offered by distributed intelligence.

Keywords: supply chain management, stock management, JADE.

1 Introduction

Work(s) concerning the application of the multi-agent approach have existed for more than a decade [3,1]. In [2], the author compared the multi-agent approach for supply-chains management to conventional approaches.

This work focuses on the problem of inventory management which is an important element of the supply-chain management problem. The goal of our project was to develop an application supporting decision making with regards to optimizing strategies for producing electrical machines. The system suggests which strategies should be used for each product in order to minimize costs while keeping short delivery time. The strategies considered in this work are called ATO (Assembly-to-Order) and MTO (Make-to-Order). ATO means that all parts that go into manufacturing electrical machine are stored in the warehouse. When an order arrives, parts are taken from the warehouse and the electrical machine is delivered as soon as it is produced. MTO strategy means that nothing is held in storage and the parts are ordered from the suppliers only when an order arrives, however the client has to wait much longer. The benefits in the case of an ATO strategy are: higher customer satisfaction and a higher number of electrical machines sold, however higher costs of storage. In the case of the MTO strategy the advantages are a lower cost of storage but also a higher risk of the customer withdrawing his order because of the lead time being too long.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 343–345, 2007.

2 Description of Model and Solution Algorithm

One way to accelerate the computations is to distribute them into several parts. The electrical machines are divided into several groups, called clusters, which can be analyzed separately. The electrical machines in one group have to be analyzed together because they use some common types of parts and the choice of production strategy for one electrical machine has a high influence on the estimation of strategies for other electrical machines within the same group. These dependencies are caused by the mutual influence on the stock levels of parts of the common types and especially on the values of the safety stocks and cycle stocks.

The model is logically distributed and consists of several computation nodes. In each computation node there are several different types of agents operating. *DataAgent* provides access to database, converts raw data into objects and divides them logically into clusters. DataAgent assigns each cluster to a different *EngineAgent*. After obtaining a cluster, each EngineAgent wakes up *TransformationAgent* and sends the cluster to it to perform computations. EngineAgent can use several TransformationAgents for the cluster which may execute different optimization algorithms so as to improve the solution space search. Currently two methods are implemented: an utter review method and a genetic algorithm. As a result we receive electrical machines with an associated strategy (ATO or MTO) and a cost for the whole cluster. Costs for ATO consist of the following elements: lost sale for given levels of client satisfaction, costs of maintaining optimal levels of safe stock for required parts and costs of maintaining optimal levels of cycle stock for required parts. MTO costs take into consideration the profits lost for the calculated delivery times of electrical machines to consumers. The strategy assignment for electrical machines in the cluster is the best, if the total cost for the electrical machines within the whole cluster is the lowest found.

3 Experimental Results

Many **comparative tests** were preformed to prove that the proposed solution gives better results than producing all electrical machines according to the ATO or to the MTO strategy. Among others, the index SCP (stocks-costs-penalties), that expresses the costs of maintaining stocks (cycle stocks and safety stocks) and penalties taking place when orders from the customer are not realized, were compared. As one can see in fig. 1a)—profits resulting from the proposed approach are obvious and very significant.

The goal of making the **efficiency tests** was to demonstrate the profits of using the agent technology in the presented project. Computations performed by *TransformationAgents* may be done not only on the server but also on the client side—i.e. some parts of the biggest clusters may be calculated on client computers. Three types of tests have been made: (i) “All local” - the server is launched on one host, three client applications run on the same host; (ii) “Client remote” - Server is separated from client application (different hosts),

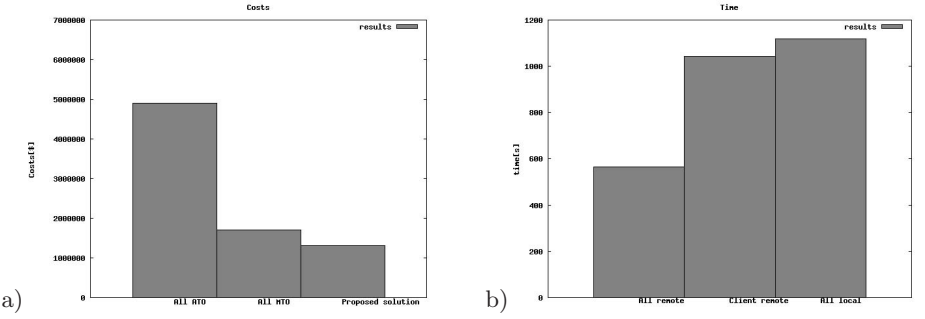


Fig. 1. Comparison of proposed and default solutions related to a) production costs SCP values for selected strategies: all electrical machines produced with ATO strategy, all electrical machines produced with MTO strategy, electrical machines produced with proposed solution, b) computation time

but computations are performed on the server side; (iii) Server is separated from client application (different hosts), computations are performed on the client side. As one may see in fig. 1b), distributing the computations to client hosts result in larger profits in time. The end user gets results much faster.

4 Conclusions and Future Works

The multi-agent approach offers advantages on both the conception and realisation levels. On the conception level it makes the identification of elements of problem domains and their description as agents simple. On the realization level, the application of the multi-agent platform JADE made implementation of agents, their cooperation as well as the distribution of the whole system possible. Future work on the system will embrace an introduction of aspects related to modeling of dynamics of the management of the production process and taking the time flow into account as well as the more developed problem domain. In the new system simulation of the whole process of creating and optimizing supply chain will be performed. Agents will cooperate with each other, perform negotiations and will be more flexible.

References

1. Moyaux, T.: Design, Simulation And Analysis Of Collaborative Strategie. In: MultiAgent Systems: The Case Of Supply Chain Management, Ph.D. Thesis, Université Laval (2004)
2. Parunak, V.: Applications of Distributed Artificial Intelligence in Industry. In: O'Hare, Jennings (eds.) Foundations of Distributed Artificial Intelligence, ch. 4, Wiley Inter-Science, Chichester (1994)
3. Shen, W., Norrie, D.H.: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and Information Systems 1(2) (1999)

The Impact of Network Topology on Trade in Bartering Networks – Devising and Assessing Network Information Propagation Mechanisms

David Cabanillas[1] and Steven Willmott[1]

Technical University of Catalonia,
Software department,
Campus Nord, Omega building
Jordi Girona Salgado, 1-3
Barcelona (08034), Spain
{dconrado,steve}@lsi.upc.edu

Abstract. Resource allocation in distributed systems is an exciting area of research. Inherent properties in this environment, such as strategic users acting selfishly and the structure of the environment within which exchanges occur, are relevant challenges to study. This paper proposes a market–based resource allocation in a distributed environment and explores the effects of network structure on the allocation of performance together. Further, we proposed mechanisms to improve the performance of the market. The proposed model, as well as mechanisms to maximize the allocation of objects/goods have been implemented and studied experimentally. The results obtained show how topology affects the performance of the market. Using information propagation mechanisms clearly contributes to its improvement.

Keywords: Market–based, Network Topology, Peer–to–Peer Computing.

1 Introduction

This paper [1] is concerned with analyzing how network topologies affect bartering processes in markets. The analysis in particular focuses on systems with:

- Sparse topologies (with few links per node).
- Self–interested agents (which only trade for immediate gain).
- Random networks.
- A variety of simple information propagation mechanisms.

The simulation parameters used to model the market are: 500 nodes with 3000 objects and an average initial level of satisfaction (*l.o.s.*) equal to 7,000 points.

In order to contrast the effect that the quantity of links has on the performance of the market, a set of scenarios where the quantity of links has been varied has been simulated: from a fully connected, s1 with 124,759 links to a quasi non-connected topology, s6 with 779 links.

[1] An extension of this work is available in the Technical Report LSI-07-22-R.

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 346–348, 2007.

2 Simple Bartering and Effects of Information Propagation on Trade Diffusion

Without propagation, in figure 1 a) s1, s2 and s3 the *l.o.s.* gets a value near to 7,800 points. In the rest of the scenarios, the value obtained is far from the optimal *l.o.s.*. But in scenarios where the average path length is greater than two, some simulations have shown that no trades are made in the market.

With propagation, in figure 1 b) at the start time (i.e. from time 0 to 15) behavior is similar to that of the previous case. In scenarios s1, s2 and s3, the *l.o.s.* is above 9,000 points. In s4 the *l.o.s.* is near to 8,500 points. When the network has 1,559 links (i.e. s5) the *l.o.s.* is around 7,500 points and in the s6 is where the value obtained is farthest from the optimal *l.o.s.*. In this case, the worst results appear when the average path length is greater than 3.

In order to improve the current results, two extensions have been implemented:

- Extension 1: Avoiding the re–sending of preferences.
- Extension 2: Promoting the propagation of preferences in nodes with few links and to promote the propagation of neighbors in nodes with many links.

Figure 2 shows the *l.o.s.* when the market is using propagation of preferences based on extension 1 and extension 2. The experiments shown here, focus on scenarios s4 and s5 as theses are the scenarios where there are significant variations in the *l.o.s.*. The first column of the set is with the market using the original propagation, the second column is the *l.o.s.* obtained with extension 1 and finally the last column is related to the extension 2. The results reveal that under 1,559 links neither extension 1 nor extension 2 work properly. The low quantity of links makes this market a sterile market, as much for the original approach as well as for the proposed extensions. In s5 with 1,559 links, the results using extension 2 are better than in the original and extension 1 approaches. In s4, both extensions improve the results of the original approach.

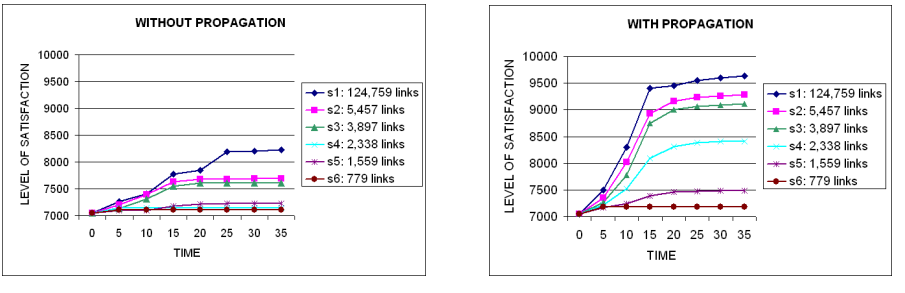


Fig. 1. a) Results working without propagation and b) Results working with propagation

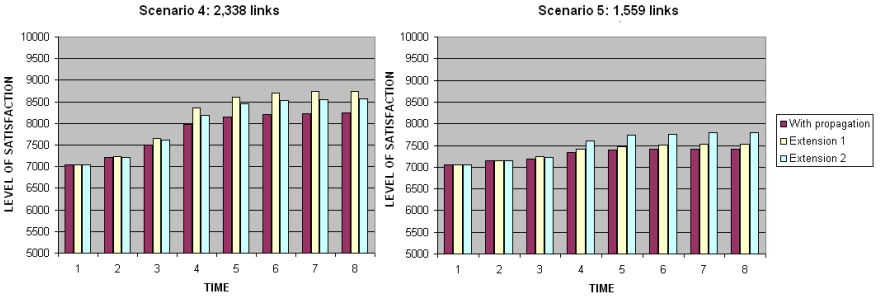


Fig. 2. *L.o.s.* with the passing of time working with original propagation, extension 1 and extension 2

3 Discussion

Whilst simple, the results show two things. The first is that the quantity of links clearly has an impact on the performance of the market. In fully connected networks, all nodes are free to trade with any node in the market. Optimal allocations are possible amongst nodes with few trades. The second is that market performance is affected by the use of propagation of preferences. Even simple propagation can significantly change the efficiency of the market.

Comparing the results working with and without the propagation mechanisms, we can establish that the asset using propagation is relevant. To use a propagation list has a computational cost and can also be a handicap that in topologies with a high density of links since propagation and analysis of preference lists costs more than the relative gain. The model applied limited the length of the preference list propagated, hence processing was still relatively rapid - however comparison studies are needed to identify the tradeoffs between propagation mechanisms and different topologies.

To summarize, the topology has a direct effect on the performance of the market. The quality and quantity of information from one individual to another, in our case propagation of preferences, has a positive influence on the performance of the market.

Author Index