Monitoring Large-Scale Multi-Agent Systems using Overhearing

Gery Gutnik
Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University
Ramat Gan, Israel
January 2006

This work was carried out under the supervision of Dr. Gal A. Kaminka, Department of Computer Science, Bar-Ilan University.

Acknowledgments

Writing this acknowledgement, I have realized that the list of people I should be thanking is quite long. Obviously, the first person I would like to express my gratitude is my advisor Dr. Gal A. Kaminka. I would like to thank Gal for the challenging research topics, motivating debates and instructive guidance. More than that, I would like to thank Gal for his warmth and kind nature, not just as an advisor, but also as a human being. Indeed, working with Gal was a pleasure. At this point, I would also like to thank Gal's wife, Oshra Kaminka, for her patience (to our late meetings) and her hospitality.

I would like to show my respect and appreciation to my Master thesis advisor, the late Dr. Yuri Zlotnikov. It was his parental guidance that convinced me to proceed to the Ph.D. studies in the first place.

Next, I would like to thank Prof. Sarit Kraus: Although Sarit was not directly involved in my thesis, her silent support (in matters concerning scholarships, publications and travels) has taken some of the pressure off my shoulders. Also, I would like to thank Dr. Onn Shehory, Dr. Ely Porat and Dr. Moshe Lewenstein for offering useful assistance and advice at different stages of my research.

Special thanks go to Dr. Meirav Hadad. I would like to thank Meirav not just for introducing me to Gal, but also for all her help throughout my Ph.D. and especially in writing this thesis.

I want to thank everyone on the Maverick group for creating such a friendly environment in the lab and also for their informative remarks and comments on my research. Also, I would like to thank Dafna and the Administrative Stuff of the Computer Science Department. Indeed, their help in administrative matters, supplementary to the research itself, eased up the long period of my studies in Bar-Ilan.

Finally, last but not least, I would like to express my appreciation to my family–my parents, my sister and her spouse, my grandmother–and all my friends for their patience, support and encouragement both in the good times and in the times of crisis.

Contents

List of Figures

List of Algorithms

Abstract

Overhearing is fast gaining attention as a generic method for monitoring open, distributed multi-agent systems. In such settings, agents' internal structure is not generally known to a monitoring agent, but overhearing does not require such knowledge. Instead, the monitoring agent uses the overheard routine communications as a basis for inference about the other agents. Our work focuses on cooperative overhearing, in which the overheard agents usually know they are being overheard, and do not in any way intend to disrupt the monitor.

Previous work on overhearing investigated an extensive set of techniques and implementations of overhearing. However, focusing mainly on its potential applications, those investigations often rely on assumptions related to the fundamentals of overhearing. In contrast, we dedicate our research to a comprehensive study of the fundamental building blocks that allow overhearing in the first place. In doing so, we systematically tackle various assumptions made by previous investigations. In particular, our study focuses on overhearing in large-scale multi-agent systems and addresses the specific challenges and limitations that characterize such settings.

The first overhearing building block, addressed by our research, is the representation of multi-agent conversations. Various formalism have been proposed for that purpose. In particular, recent investigations showed Petri nets to provide a viable representation approach for modelling multi-agent interactions. By analyzing the strengths and weaknesses of the rather radical Petri net approaches introduced by previous work, we propose a novel representation technique especially suitable for overhearing. Furthermore, we show this representation to be more scalable than previous representations, and thus more appropriate for monitoring conversations in large-scale settings. We show that this new representation offers a comprehensive coverage of essentially all conversation features needed to represent complex multi-agent conversations. We also present a procedure for transforming human-readable AUML conversation protocol diagrams to our machine-readable Petri net representation.

Next, we addressed the building block of conversation recognition. Conversation recognition is the process of identifying the actual conversation based on a sequence of overheard messages. In the process, the overhearing agent extracts various parameters of the overheard conversation such as the set of conversing agents, the corresponding conversation protocol, etc. In addition, it also handles possible errors caused by differences between the conversation as it was overheard and as it was

actually carried out by the agents (e.g., in cases where the overhearer was not able to overhear some of the exchanged messages).

Although conversation recognition is a key step in overhearing prior to any possible inference based on overheard communications, it is often discarded by previous investigations. Our work addresses the challenges related to conversation recognition by first introducing a formal model of overhearing. Most previous works focus on potential applications of overhearing. Therefore, the proposed model was the first to formalize the general problem of overhearing unrelated to any specific task.

Then, based on this model, we provide a skeleton algorithm for conversation recognition, and provide instantiations of it for lossless and lossy settings. Since in large-scale multi-agent systems overhearing agent has to process large quantities of intercepted messages, conversation recognition algorithms must be efficient. Accordingly, the time-complexity of these algorithms was analyzed. We show that handling conversation recognition of systematic message loss, which is unique to overhearing, is significantly more efficient than handling the general case of randomly lost messages (which is intractable).

The final building block addressed in this work is selective overhearing, i.e. overhearing under the restriction of selectivity. The restriction of selectivity is mainly compelled by the specific characteristics of large-scale multi-agent systems. In such settings, it is reasonable to assume that the overhearing resources will be essentially limited, thus allowing the overhearing agent to overhear only a subset of inter-agent communications carried out in the monitored settings. Accordingly, the overhearer must be careful in choosing which targets to overhear on account of other potential targets.

Most previous investigations on overhearing ignore the limitation of selectivity, assuming that all relevant inter-agent communications can be overheard. Tackling this problematic assumption, our work provides an empirical study of selective overhearing focusing on widely common hierarchical organizations. Here, we first propose a model for selective overhearing of such organizations. Then, using a simulation of this model, we perform an extensive set of experiments in which we empirically evaluate and compare performance of various overhearing policies taking into consideration both the limitations of selectivity and the specific characteristics of hierarchical organizations.

We empirically study both centralized and distributed selective overhearing policies. In doing so, we tackle another problematic assumption by previous investi-

gations. Those investigations either assume a single overhearer or a group of non-cooperative overhearing agents that perform overhearing out of their own interest. In contrast, our work considers overhearing committed by a group of collaborative overhearers. First, we consider centrally-coordinated overhearing groups which are equivalent to a single centrally-located overhearer. But, then, we empirically study the transition to a group of overhearing agents acting collaboratively in a distributed manner.

Based on the performed experiments, we were able to isolate the factors influencing the behavior of those policies and reach several qualitative conclusions. With respect to centralized overhearing policies, we have found a classical value-volume tradeoff. This tradeoff was found to be surprisingly robust to many characteristics of hierarchical organizations. However, what was more surprising is that the combination of two types of policies (value and volume) in addition to being fully robust, outperformed each of the policies separately.

Addressing distributed policies, we considered the transition from effective centralized policies to distributed ones by gradually decreasing the level of collaboration between the overhearing agents. Here, we found some factors to significantly influence the performance of the examined policies, while finding that others can simply be neglected or only partially solved.

Chapter 1

Introduction

Over recent years, we have witnessed an increasing interest in *multi-agent systems* (MAS). In particular, *open, distributed* MAS have gained much attention. In such settings, multiple agents often perform mutually-dependent tasks. These agents are geographically distributed and independently built, and may thus be heterogeneous in terms of design. Consequently, in order to allow each MAS application designer to focus on the design of its agents, the coordination of the performed activities is often accomplished through inter-agent communications.

1.1 Monitoring in Multi-Agent Systems

One of the key challenges in MAS applications has always been, and still remains, the task of monitoring. The goal of monitoring is to provide information about the monitored agents. In turn, this information can further be used for a variety of tasks such as performance analysis [Turner and Jennings, 2000], adaptation to changes [Guessoum *et al.*, 2004], diagnosis [Kalech and Kaminka, 2005], visualization and debugging [Ndumu *et al.*, 1999] and so on.

Monitoring can be done both by an external operator (human or agent) or by one of the application agents. An external monitor can be used, for example, to ascertain the correct and effective execution of MAS application. Using the gathered information, the monitor may act based on the state of the monitored system. Its actions can either be reactive, e.g., in adapting to changes in the environment [Guessoum *et al.*, 2004], or proactive, such as when identifying opportunities for offering assistance [Aiello *et al.*, 2001, Busetta *et al.*, 2001, Busetta *et al.*, 2002].

Similarly to the external monitor, an application agent is often required to monitor its environment and peers. In most MAS applications, and in particular those that involve teamwork, an application agent can not solely achieve its goal. Achieving their goals, application agents often inter-depend on other agents and their collaboration [Cohen and Levesque, 1991, Jennings, 1993, Parker, 1993, Grosz and Kraus, 1996, Tambe, 1997, Kaminka and Tambe, 2000]. Thus, application agents monitor their peers using them as information sources, finding opportunities and needs for collaborations, etc [Kaminka, 2000].

1.2 Monitoring via Overhearing

The distinct characteristics of open distributed multi-agent settings make it difficult to monitor MAS applications. Various monitoring techniques have been developed [Jennings, 1993, Tambe, 1997, Klein and Dellarocas, 1999, Kaminka *et al.*, 2002].

The traditional technique is a *report-based* monitoring [Jennings, 1993, Tambe, 1997, Klein and Dellarocas, 1999]. It requires application agents to report their status to the monitor agent under a predefined set of conditions (e.g. at a regular interval, whenever an agent changes its state, etc). However, this monitoring technique is not suitable for open distributed multi-agent systems [Kaminka *et al.*, 2002]. First, the reports may require a significant overhead in communications, and, thus, affect the agents' performance. Second, report-based monitoring requires compliance by all agents to support the variety of reports that may be needed. The continued maintenance of such capability is not feasible in large-scale settings and particularly not in settings where the agents can be built by unknown sources.

Fortunately, routine inter-agent communications present a basis for a promising alternative. Given the ongoing standardization in inter-agent communications, an opportunity exists for monitoring agents by overhearing their conversations. The work by [Kaminka *et al.*, 2002] introduced this technique referring to it as *monitoring by overhearing*. A schematic description of monitoring by overhearing is shown in Figure 1.1[1].

Using monitoring by overhearing, an overhearing agent listens to the routine inter-agent communications and overhears exchanged communication messages be-

[1]Although in Figure 1.1 the overhearing agent is shown as an external monitor, the overhearer can be one of the application agents as well.

tween application agents. The latter usually know that they are being overheard but do not care. This is the case of cooperative overhearing studied by all previous investigations on overhearing. Here, the communicating agents are assumed to be cooperative with the overhearer by not trying to disguise their routine communications. Similarly to those previous works, this thesis assumes that the monitored agents do not in any way intend to disrupt the monitor from overhearing their communications.

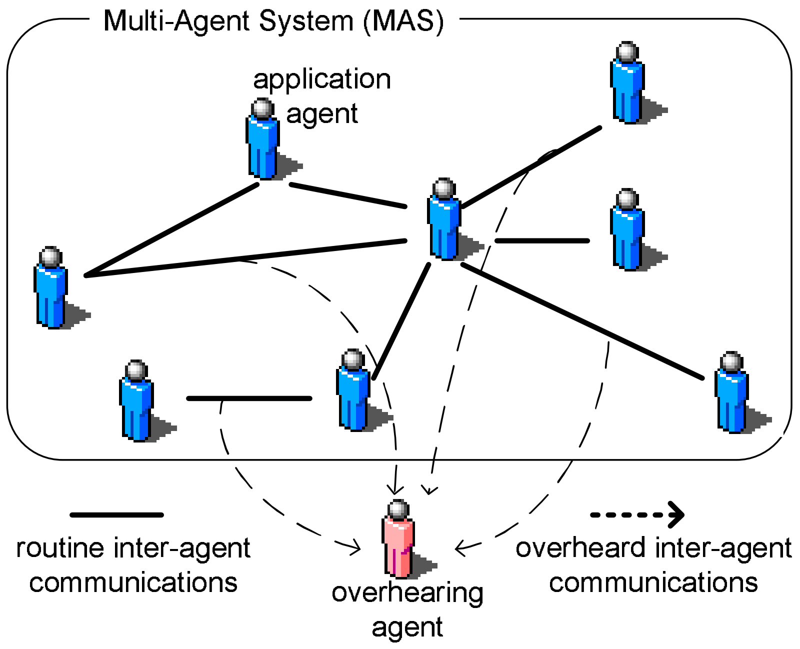

Multi-Agent System (MAS)
application agent
routine inter-agent communications
overhearing agent
overheard inter-agent communications

Figure 1.1: Overhearing Concept.

Given the cooperative assumption, it can be claimed that more simple and direct solutions can be used. Specifically, communicating agents can actively assist the monitor to gather necessary information (e.g., by broadcasting specifics of their communications). However, this sort of collaboration brings back the drawbacks of the *report-based* monitoring discussed at the beginning of this section. On the other hand, cooperative overhearing solely requires an assistance by neutrality, i.e. not caring that the communications are being overheard. In turn, the overhearing agent uses the observed communications to independently assemble and infer the monitoring information. Accordingly, monitoring via overhearing does not require any changes in the behavior of application agents and thus is more suitable for open distributed multi-agent systems.

Overhearing is a generic monitoring technique that relies only on the ability to overhear inter-agent communications. Thus, just as with general monitoring discussed in the previous section, overhearing can be done both by an external monitor [Aiello *et al.*, 2001, Busetta *et al.*, 2001, Kaminka *et al.*, 2002, Poutakidis *et al.*, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005] and by an application agent that overhears it peers [Novick and Ward, 1993, Legras, 2002].

1.3 Motivation: Overhearing Building Blocks

Previous investigations on overhearing have demonstrated a range of overhearing techniques as summarized by Figure 1.2. The bulk of existing work addresses applications implemented based on inference made from overheard routine inter-agent communications. These applications include maintaining situational and organizational awareness [Novick and Ward, 1993, Legras, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005], monitoring progress [Kaminka *et al.*, 2002], debugging and detecting inconsistencies [Poutakidis *et al.*, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005] and discovering opportunities for providing advice [Aiello *et al.*, 2001, Busetta *et al.*, 2001]. A detailed discussion of these investigations is provided in Chapter 2, Section 2.1. However, focusing on specific implementations, these investigations leave open many of the overhearing fundamentals.

In contrast to these investigations, our research focuses on the fundamental building blocks that enable these overhearing applications in the first place. Those building blocks tackle different problematic assumptions made by overhearing applications, and their implementation is a key step in allowing overhearing. The investigations on overhearing building blocks include our work in this thesis (Figure 1.2 also contains citations of our previous publications as shown later in Section 1.5) and works by [Busetta *et al.*, 2002, Platon *et al.*, 2004, Platon *et al.*, 2005, Fan and Yen, 2005]. In this section we provide a brief explanation of the concept behind each of the overhearing building blocks, while a detailed discussion on their related work is found in Chapter 2, Sections 2.2-2.5.

Figure 1.2 above summarizes the four overhearing building blocks addressed in literature. These building blocks are (i) overhearing communications; (ii) representing conversations; (iii) conversation recognition and (iv) selective overhearing. Below, we provide a discussion on each of these building blocks.

Overhearing building block (i): ***Overhearing communications***. Accessibility of relevant inter-agent communications is crucial for monitoring and inference based on overhearing. Accordingly, overhearing applications assume that the com-

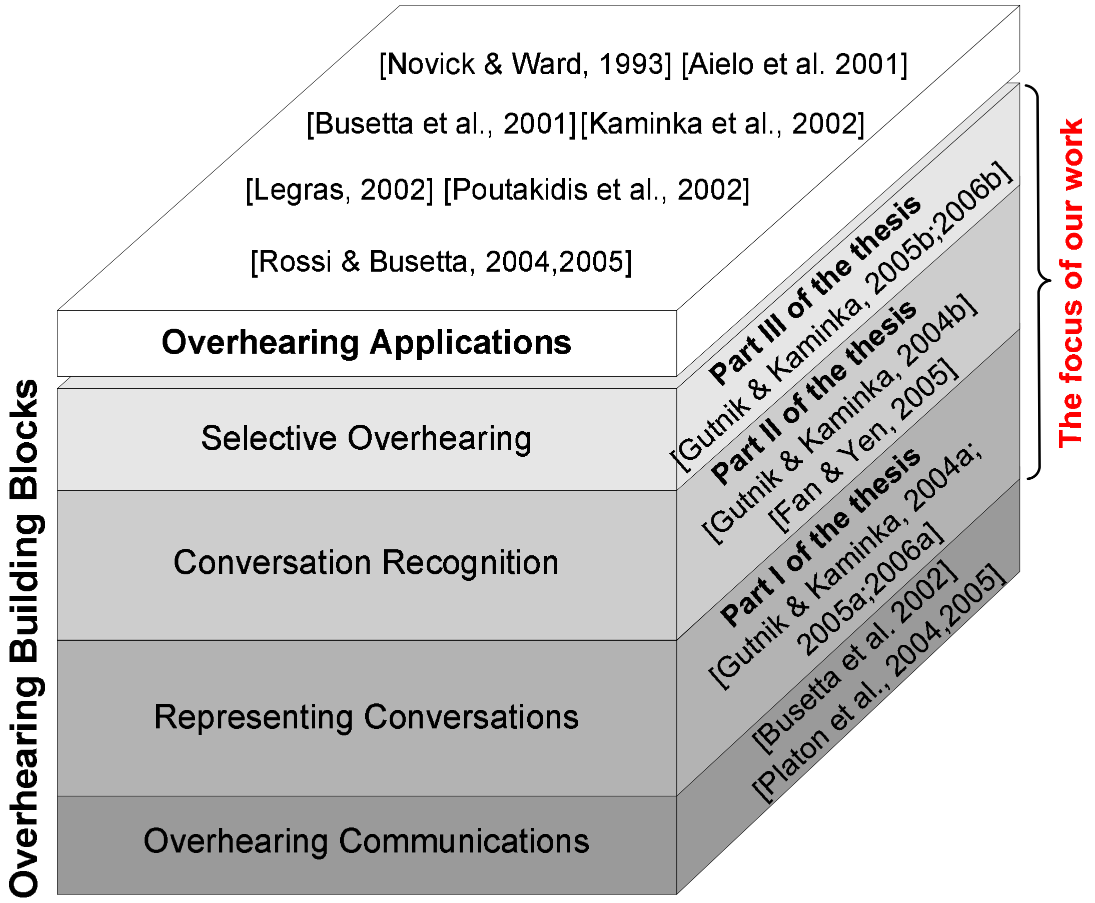

[Novick & Ward, 1993] [Aielo et al. 2001]
[Busetta et al., 2001][Kaminka et al., 2002]
[Legras, 2002] [Poutakidis et al., 2002]
[Rossi & Busetta, 2004,2005]
Overhearing Applications
Selective Overhearing
Conversation Recognition
Representing Conversations
Overhearing Communications
Overhearing Building Blocks
Part III of the thesis
[Gutnik & Kaminka, 2005b;2006b]
Part II of the thesis
[Gutnik & Kaminka, 2004b]
[Fan & Yen, 2005]
Part I of the thesis
[Gutnik & Kaminka, 2004a; 2005a;2006a]
[Busetta et al. 2002]
[Platon et al., 2004,2005]
The focus of our work

Figure 1.2: Overview of Related Work on Overhearing.

municative acts exchanged between overheard agents can be obtained. The investigations, addressing this building block, consider different techniques for obtaining exchanged communications.

Overhearing building block (ii): ***Representing conversations.*** Overhearing relies on routine inter-agent conversations carried out by the overheard agents. Listening to the conversations between the monitored agents and tracking them, the overhearer must be able to represent those overheard conversations.

Overhearing building block (iii): ***Conversation recognition.*** Being able to represent conversations and to obtain exchanged communicative acts, overhearing agent uses these capabilities to perform conversation recognition. Conversation recognition is a key step in overhearing often discarded by previous investigations. It enables overhearer to identify the actual conversations carried out in the monitored settings based on a set of overheard messages. In the process, the overhearing agent identifies various parameters of overheard conversations (e.g. a set of conversing agents, its corresponding conversation protocol, etc) and handles problems caused by possible differences between the actual conversation between agents and the conversation as it was overheard by the monitor (e.g. losses).

Overhearing building block (iv): *Selective overhearing.* Another problematic assumption, made by existing applications using overhearing, is the ability to overhear *all* inter-agent communications. This assumption is often challenged in real-world settings, and in particular, in *large-scale* MAS. Instead, the overhearing agent has limited resources, and may only overhear a subset of conversations carried out in monitored settings. Consequently, the overhearer must be selective in choosing which conversations must be overheard.

1.4 Challenges Addressed & Contributions

In our research, we focus on overhearing building blocks (ii)-(iv). We attempt to systematically cover various aspects related to these building blocks, tackling problematic assumptions made by previous investigations.

In doing so, we particularly focus on overhearing *large-scale* settings nowadays widely used both in multi-agent community and the real-world. Overhearing large-scale MAS raises a variety of specific challenges caused by the size, the large number of agents and conversations to overhear and other factors.

Contributions with respect to building block (ii):
Scalable Conversation Representation for Overhearing

Overhearing is based on monitoring routine inter-agent communications. To monitor those communications successfully, we must first be able to represent multi-agent conversations. Addressing this general challenge (which is important not only in context of overhearing), various formalisms and techniques have been proposed, each having its own advantages and weaknesses [Smith and Cohen, 1996, Parunak, 1996, Odell *et al.*, 2000, AUML site, 2005].

In particular, Petri nets [Petri Nets site, 2005] have been found useful in representing multi-agent conversations [Cost, 1999, Nowostawski *et al.*, 2001, Purvis *et al.*, 2002, Mazouzi *et al.*, 2002]. However, even here, rather different approaches have been introduced [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Nowostawski *et al.*, 2001, Purvis *et al.*, 2002, Cranefield *et al.*, 2002, Mazouzi *et al.*, 2002, Poutakidis *et al.*, 2002]. We classify and analyze these previous Petri net approaches for representing multi-agent conversations.

Based on the insights gained from this analysis, we propose a novel Petri net representation which has two major advantages with respect to previous works. First, it is particularly suitable for monitoring by overhearing. Second, and more important, it is more scalable in terms of the number of agents, the number of exchanged messages and the number of concurrent conversations that can be represented. Thus, it is especially appropriate for representing multi-agent communications in large-scale settings.

Using the proposed representation, we addressed two additional challenges left by previous investigations. First, we systematically covered a variety of conversation aspects needed to represent complex multi-agent interactions. We have shown how this representation can be used to represent essentially all features of FIPA AUML conversation standards, including simple and complex interaction building blocks, communicative act attributes such as message guards and cardinalities, nesting, and temporal aspects such as deadlines and duration. Although some of those have been addressed before, no previous work showed a comprehensive coverage of all these aspects.

Finally, as our last contribution in context of conversation representation, we formalized the conversion of AUML protocol diagrams to their equivalent Petri nets representation. Even though AUML is a widespread human-readable representation, it can not be used in automatic procedures of debugging [Poutakidis *et al.*, 2002], validation [Desel *et al.*, 1997], deadlock detection [Khomenco and Koutny, 2000], etc. In contrast, these can be applied on machine-readable Petri net representations. Our work takes the first steps in this direction providing a semi-automatic procedure for transforming conversation protocols in AUML to their Petri net form.

Contributions with respect to building block (iii):

Formal Model & Algorithms for Conversation Recognition

The next research topic addressed in our work was the topic of conversation recognition based on the representation of inter-agent communications studied earlier. Conversation recognition is a key step in monitoring via overhearing. Furthermore, it is a preliminary step to any further inference possible based on overheard information. Overhearing inter-agent communications, an overhearing agent has access only to separate communicative acts exchanged between the monitored agents.

However, the actual conversations and their different parameters are still have to be recognized and extracted from the sequence of overheard messages. This process is called conversation recognition.

With respect to the process of conversation recognition, some key assumptions made by previous works on overhearing are difficult to extract. For instance, all previous investigations assume that the overhearing agent can match intercepted messages to a conversation protocol. Most make the assumption that *all* messages in a conversation are overheard (i.e. no losses). Yet both assumptions can be challenged in real-world settings. Our work on conversation recognition tackles these problematic assumptions.

Furthermore, all previous investigations addressed overhearing in context of specific applications without any formalization of the general problem. Given that, our first contribution was to provide a comprehensive formal model of overhearing unrelated to any specific task.

As our second contribution, we have formalized the problem of conversation recognition using the proposed model. Then, based on this formalization, we provided a set of algorithms solving the problem of conversation recognition for both lossless and lossy settings. Since in large-scale multi-agent systems overhearing agent has to process large quantities of intercepted messages, conversation recognition algorithms must be efficient. Accordingly, the time-complexity of these algorithms was analyzed. We have shown that handling the general lossy case is computationally hard. On the other hand, handling a more specific case of losses, common in monitoring via overhearing, is computationally possible.

Contributions with respect to building block (iv):
An Empirical Study of Selective Overhearing in Multi-Agent Organizations

In our research on conversation recognition, we addressed the cases where some messages of overheard conversations might not be accessible to the overhearing agent (due to noise, overhearer's position, etc). In the final research topic, addressed in this thesis, we consider the case where complete conversations are unknown to the overhearing agent, rather than messages.

In large-scale MAS settings, it is reasonable to assume that overhearing resources are bound to be limited. Accordingly, the assumption that *all* relevant inter-agent communications can be overheard, made by all previous investigations on overhearing, is problematic in such settings. Instead, an overhearing agent must be selective in choosing its targets. Thus, while being able to overhear some of the conversations, others will necessarily remain unknown to the overhearer.

Under the restriction of selectivity, the main challenge is to effectively choose overhearing targets from all the potential ones. Our work reports on an empirical study of selective overhearing, making these choices based on organizational knowledge of the monitored settings. We focus on hierarchical organizations, which are both common in multi-agent applications [So and Durfee, 1996, Yadgar *et al.*, 2003] and in real-world corporates.

We first propose a theoretical model of selective overhearing in hierarchical organizations. This model extends the formal model of overhearing proposed in context of conversation recognition with respect to aspects of selectivity and specific characteristics of hierarchical organizations [Dewan *et al.*, 1997, Gannon and Newman, 2001, Best *et al.*, 2003, Jensen, 2003, Friebel and Raith, 2004].

Based on this model, we simulate inter-agent communications in hierarchical organizations and empirically evaluate and compare various selective overhearing policies. Both centralized and decentralized selective policies are studied. In contrast to previous investigations on overhearing, which either assume a single overhearing agent or a group of self-interested non-collaborating overhearers, we study overhearing performed by a group of cooperating overhearing agents. First, we consider centrally-coordinated overhearing groups which are equivalent to a single centrally-located overhearer. But, then, we empirically study the transition to a group of overhearing agents acting collaboratively in a distributed manner.

Our experiments on centralized selective overhearing policies showed a classical value-volume tradeoff. This tradeoff was found to be surprisingly robust to many characteristics of the monitored organizations. Further studying centralized policies, we have come to another surprising conclusion: Combining the two types of policies (such that some agents follow the value type policy, while others follow the volume type policy) improves their individual performance. Moreover, the combined policies have been found to be effective unrelated to any characteristic of the monitored organization (even those that influence each policy separately).

Then, we have studied the transition from centralized to distributed selective overhearing policies. We identified the various factors influencing the coordination of the overhearing agents in effective centralized policies, and considered the effects of decreasing their inter-dependence with respect to those collaboration factors.

Here, we found that the various collaboration factors differ significantly in their influence on the performance of the examined policies. In particular, we have found that the decrease in visibility–knowledge on where and when the monitored agents converse–causes a significant degradation in the policies' performance. In contrast, the use of the highly-maintained shared memory has been found to have no effect over the use of an individual memory which is easily maintained. The last factor considered was collision avoidance, i.e. avoiding the situations where the same target overheard by two or more overhearers. We have found that this time consuming activity can be only partially solved.

1.5 Publications

Subsets of the results that appear in this dissertation were published in the proceedings of the following refereed journals, conferences, books and workshops:

Conversation Representation for Overhearing:

- [Gutnik and Kaminka, 2006a] Gutnik, G. and Kaminka, G. A. Representing Conversations for Scalable Overhearing. *Journal of Artificial Intelligence (JAIR)*. To Appear, 2006.

- [Gutnik and Kaminka, 2005a] Gutnik, G. and Kaminka, G. A. A Scalable Petri-Net Representation of Interaction Protocols for Overhearing. In van Eijk, R. M., Huget, M.P. & Dignum, F. (Eds.), *Agent Communication, LNAI vol. 3396*, pages 50–64. Springer-Verlag, 2005. An earlier version appeared in the AAMAS-04 Workshop on Agent Communications.

- [Gutnik and Kaminka, 2004a] Gutnik, G. and Kaminka, G. A. A Scalable Petri-Net Representation of Interaction Protocols for Overhearing. In *Proceedings of AAMAS-04 (poster)*, pages 1246–1247, 2004.

Formal Model & Conversation Recognition:

- [Gutnik and Kaminka, 2004b] Gutnik, G. and Kaminka, G. A. Towards a Formal Approach to Overhearing: Algorithms for Conversation Identification. In *Proceedings of AAMAS-04*, pages 78–85, 2004.

Selective Overhearing:

- [Gutnik and Kaminka, 2006b] Gutnik, G. and Kaminka G. A. Experiments in Selective Overhearing of Hierarchical Organizations. In van Eijk, R. M., Flores, R. & Huget, M. P. (Eds.), *Agent Communication, LNAI Book, In Press*, Springer-Verlag, 2006. An earlier version appeared in the AAMAS-05 Workshop on Agent Communications.

- [Gutnik and Kaminka, 2005b] Gutnik, G. and Kaminka, G. A. An Empirical Study of Selective Overhearing in Hierarchical Organizations. In *Proceedings of the IJCAI-05 Workshop on Modelling Others from Observations (MOO-05), Edinburgh, Scotland*, 2005.

1.6 Thesis Overview

This dissertation is constructed of eleven chapters and one appendix, where the core chapters of the thesis are organized in three main parts (see Figure 1.3). This chapter constitutes the introduction to this thesis, while the next chapter surveys the related work.

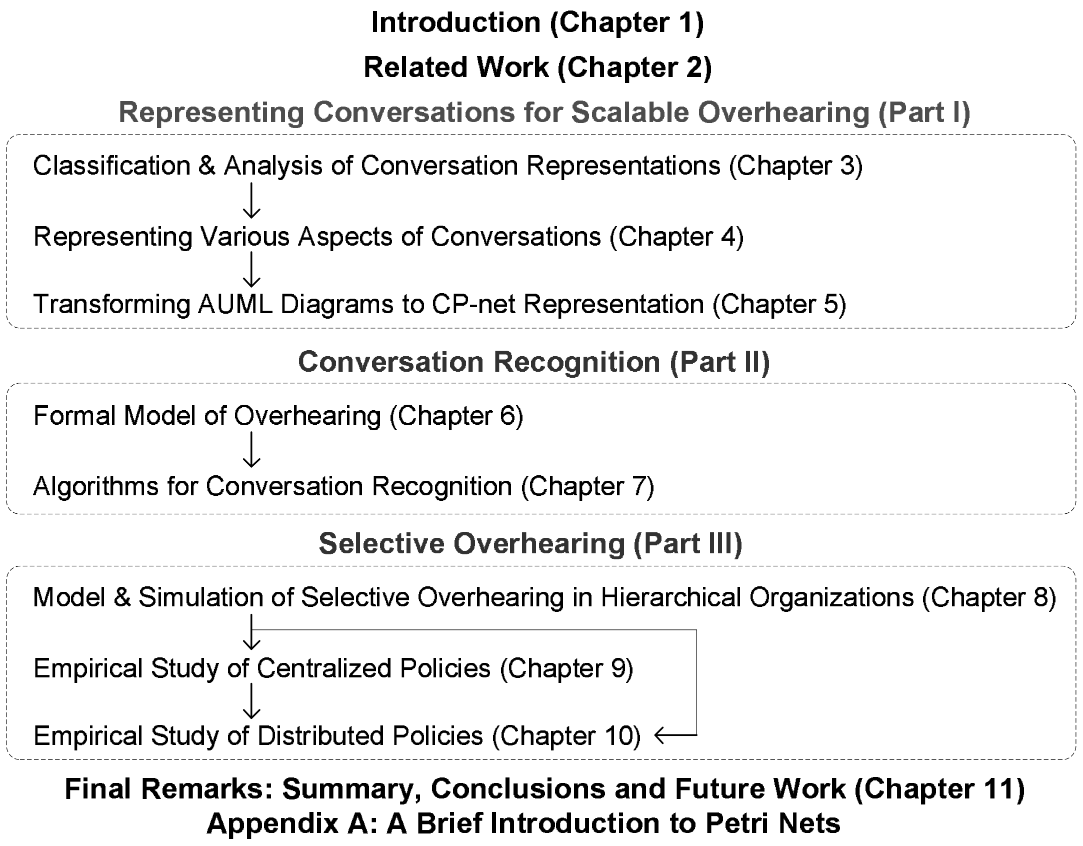

Introduction (Chapter 1)
Related Work (Chapter 2)
Representing Conversations for Scalable Overhearing (Part I)
Classification & Analysis of Conversation Representations (Chapter 3)
Representing Various Aspects of Conversations (Chapter 4)
Transforming AUML Diagrams to CP-net Representation (Chapter 5)
Conversation Recognition (Part II)
Formal Model of Overhearing (Chapter 6)
Algorithms for Conversation Recognition (Chapter 7)
Selective Overhearing (Part III)
Model & Simulation of Selective Overhearing in Hierarchical Organizations (Chapter 8)
Empirical Study of Centralized Policies (Chapter 9)
Empirical Study of Distributed Policies (Chapter 10)
Final Remarks: Summary, Conclusions and Future Work (Chapter 11)
Appendix A: A Brief Introduction to Petri Nets

Figure 1.3: Thesis Structure.

The core chapters of the dissertation, Chapters 3 to 10, are organized in three parts. Each part corresponds to our work with respect to a separate overhearing building block as presented in Section 1.4. The first part of the thesis (Part I) addresses our Petri net representation for scalable overhearing. Its chapters discuss different contributions by our research in context of conversation representation. Chapter 3 presents the classification and analysis of existing conversation representations based on which we propose our novel representation approach. We show our representation to be especially suitable for overhearing and more scalable than previously proposed approaches, and thus more appropriate for representing conversations in large-scale settings. Using this representation, we show, in Chapter 4, a comprehensive coverage of various aspects needed to represent complex multi-agent conversations. Finally, Chapter 5 introduces the semi-automatic conversion of human-readable AUML protocol diagrams to their machine-readable Petri net form.

Part II of the thesis presents our work related to conversation recognition. Chapter 6 of this part presents the proposed formal model of overhearing. Then, Chapter 7 uses this model to formalize the problem of conversation recognition and provide a set of algorithms addressing this problem. The complexity of these algorithms is analyzed to determine their appropriateness for conversation recognition in large-scale settings.

In the third part of the thesis (Part III) the problem of selective overhearing is addressed. As mentioned, the problem of selectivity rises while overhearing large-scale settings. Our work empirically studies overhearing under the restriction of selectivity of hierarchical organizations. Accordingly, in Chapter 8, we propose a model for overhearing such organizations taking into consideration limitations compelled by selectivity and specific characteristics of hierarchical organizations. Based on this model, we simulate inter-agent communications in hierarchical organizations and empirically evaluate and compare various centralized (Chapter 9) and distributed (Chapter 10) selective overhearing policies.

Rounding up, Chapter 11 presents the final remarks to this thesis providing a brief summary of our work, the main conclusions and possible directions for future research. Finally, in Appendix A, we provide a brief introduction to Petri nets which are the main target of our research in Part I.

Chapter 2

Related Work

Section 1.3 of the previous chapter presented a brief overview of related work on overhearing in order to clarify the contributions by our work as shown then in Chapter 1, Section 1.4. In this chapter, we present a detailed discussion on related work following the division shown in Figure 1.2 (see Chapter 1, Section 1.3). Accordingly, Section 2.1 presents previous investigations on overhearing applications, while Sections 2.2-2.5 show the related work on different overhearing building blocks.

2.1 Overhearing Applications

This section discusses in details previous work on applications of monitoring via overhearing. Focusing on overhearing building blocks, our work is fundamental to all of these investigations.

[Novick and Ward, 1993] show an early use of cooperative overhearing to model interactions between pilots and air-traffic controllers. In this model, pilots maintain mutuality of information with the controller not only by dialogue, but also by listening to the conversations of other pilots. While each pilot and controller act cooperatively, the other pilots are not necessarily collaborating on a joint task. Rather, they use overhearing to maintain their situational awareness out of their own self-interest.

Similarly, [Legras, 2002] uses overhearing as a method that allows agents to maintain organizational knowledge. In this approach, agents broadcast changes in their organizational memberships. Other agents use this information to track these changes and maintain organizational awareness.

[Rossi and Busetta, 2004, Rossi and Busetta, 2005] also apply overhearing for inferencing organizational knowledge. Here, overhearing agent monitors changes in

MAS settings caused by transition from one state to another. The information about agents' state, together with overheard communications, is used to deduce the organizational roles of monitored agents based on a predefined set of social rules. However, social knowledge is not the sole use of overhearing in this research. Tracking exchanged messages and state transitions, overhearer detects suspected inconsistencies and possible losses. This information is used to alert communicating agents and initiate recovery if necessary.

The latter use of overhearing is also investigated in [Poutakidis *et al.*, 2002]. Here, overhearing is applied for debugging. Using a Petri net representation of inter-agent interactions (which we further discuss in Section 2.3), the authors use overheard messages to detect failures in conversations between agents.

In contrast, investigations by [Aiello *et al.*, 2001, Busetta *et al.*, 2001] describe collaborative settings in which the overhearing agent may act on overheard messages to assist the communicating agents. The settings they describe involve communicating agents, who are engaged in problem solving. An overhearing agent monitors their conversations, and offers expert assistance.

[Kaminka *et al.*, 2002] used plan recognition in overhearing a distributed team of agents, which are collaborating to carry out a specific task. Knowing the plan of this task and its steps, the monitor uses overheard messages as clues for inferring the state of different team-members. The authors presented a scalable probabilistic representation (together with associated algorithms) supporting such inference, and showed that knowledge of the conversations that take place facilitates a significant boost in accuracy.

2.2 Overhearing Communications

Overhearing applications use routine inter-agent communications as a basis for inference. Thus, they assume that messages, exchanged between agents, can be obtained. [Novick and Ward, 1993] assume pilots to overhear communications between other pilots and air-controller by listening to a known radio-frequency. [Kaminka *et al.*, 2002] assume obtaining exchanged messages by eavesdropping.

On the other hand, a more collaborative approach was proposed by other investigations [Legras, 2002, Busetta *et al.*, 2002, Platon *et al.*, 2004, Platon *et al.*, 2005]. The work by [Legras, 2002] uses broadcast assuming overhearer to be one of the designated recipients. However, in case overhearing agent is an external listener, broadcast can be problematic [Platon *et al.*, 2005].

As a solution, a mediated approach to overhearing was proposed. In the work by [Busetta *et al.*, 2002], communications between agents are carried out through communication channels. Overhearing agents, as do other agents, have the ability to register to specific channels and receive all messages communicated through them, no matter to whom those messages are addressed.

In contrast, [Platon *et al.*, 2004, Platon *et al.*, 2005] use an active environment as a mediator. Here, all communications are compelled to pass through the environment, which in turn, decides the targets of the corresponding communications. Thus, a message, in addition to its intended recipients, can be also forwarded to an overhearer if necessary.

Throughout this thesis, we also assume that the exchanged messages can be obtained. Thus, building on top of this building block, our work is consistent with any of the introduced approaches.

2.3 Representing Conversations

The importance of modelling inter-agent communications is not limited to the scope of overhearing alone. Instead, their representation is essential in open, distributed multi-agent systems. In such settings, agents' coordination is often accomplished using standardized communications. Thus, representations of inter-agent communications can be used for visualization [Ndumu *et al.*, 1999], automated analysis [Turner and Jennings, 2000], validation and verification [Desel *et al.*, 1997], online monitoring [Kaminka *et al.*, 2002], etc.

The multi-agent community has been investing a significant effort in developing standardized *Agent Communication Languages* (ACL) to facilitate sophisticated multi-agent systems [Finin *et al.*, 1997, Kone *et al.*, 2000, ChaibDraa, 2002, FIPA site, 2005]. Such standards define communicative acts, and on top of them, *interaction protocols*, ranging from simple queries as to the state of another agent (e.g. *FIPA Request Interaction Protocol* [FIPA Specifications, 2005e]), to complex negotiations by auctions or bidding on contracts (e.g. *FIPA Contract Net Interaction Protocol* [FIPA Specifications, 2005b]). Using these interaction protocols, agents carry out conversations with each other and coordinate their activities.

Various formalisms have been proposed to describe ACL interaction protocols including *Finite State Machines* (FSM) [Smith and Cohen, 1996], *Dooley Graphs* [Parunak, 1996], AUML [Odell *et al.*, 2000, Odell *et al.*, 2001b, AUML site, 2005], etc. In particular, AUML–*Agent Unified Modelling Language*–is currently used in the FIPA-ACL standards [Odell *et al.*, 2001a, FIPA site, 2005]. In addition, UML 2.0 [AUML site, 2005], a new emerging standard based on AUML, has the potential to become FIPA-ACL standard in the future. However, a large set of FIPA specifications remain currently formalized using AUML.

Lately, there is increasing interest in using Petri nets [Petri Nets site, 2005] for representing multi-agent interaction protocols [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Nowostawski *et al.*, 2001, Purvis *et al.*, 2002, Poutakidis *et al.*, 2002, Cranefield *et al.*, 2002, Mazouzi *et al.*, 2002]. These Petri net models are extremely important in view of a broad literature on using Petri nets to analyze the various aspects of conversations in open, distributed multi-agent systems. These aspects include dynamic interpretation of interaction protocols [Cranefield *et al.*, 2002, de Silva *et al.*, 2003], validation and testing [Desel *et al.*, 1997], automated debugging and monitoring [Poutakidis *et al.*, 2002], interaction protocols refinement allowing modular construction of complex conversations [Hameurlain, 2003] and modelling agents behavior induced by their participation in a conversation [Ling and Loke, 2003].

Our work, in Part I of this thesis, also uses Petri nets to represent conversations with respect to overhearing. The proposed Petri net representation stems from a comprehensive analysis and comparison of the broad literature on using Petri nets for modelling multi-agent conversations (see Part I, Chapter 3). Accordingly, we dedicate the remaining part of this section for a detailed discussion of these previous investigations. In this discussion, we use Petri net terminology assuming the reader to be familiar with the Petri net formalism. Nevertheless, we also refer the reader to a brief introduction to Petri nets provided in Appendix A.

We start with publications by [Purvis *et al.*, 2002, Cranefield *et al.*, 2002, Ramos *et al.*, 2002]. In this approach, the authors provide a separate Petri net for each conversing agent. Thus, in order to represent a complete conversation, these different Petri nets are coordinated using a synchronization mechanism.

[Purvis *et al.*, 2002, Cranefield *et al.*, 2002] use separate *Colored Petri nets* (CP-nets) to represent different agents involved in a conversation. Here, places denote both interaction messages and states, while transitions represent operations performed on the corresponding communicative acts such as send, receive, and process. Special in/out places are used to pass net tokens between the different CP-nets, through special get/put transitions, simulating the actual transmission of the corresponding communicative acts.

[Ramos *et al.*, 2002] propose to use *hierarchy graphs* (see hierarchical CP-nets in Appendix A) for modular representation of multi-agent conversations. Different agents are modelled using separate hierarchy graphs, in which nodes denote fragments of represented conversations. These separate hierarchy graphs are synchronized using *fusion places*–a single conceptual place drawn on different Petri nets.

The works by [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Mazouzi *et al.*, 2002] presented a similar Petri net approach. Again, each conversing agent is initially modelled separately. However, the separate Petri nets are then connected into a single net corresponding to the entire multi-agent conversation.

[Cost *et al.*, 1999, Cost *et al.*, 2000] used CP-nets for representing KQML and FIPA interaction protocols. In this approach, transitions represent message events, and CP-net features, such as token colors and arc expressions, are used to represent AUML message attributes and sequence expressions. The authors also point out that deadlines (a temporal aspect of interaction) can be modelled, but no implementation details are provided. [Cost, 1999] also proposed using hierarchical CP-nets to represent complex multi-agent conversations.

In [Lin *et al.*, 2000], transitions are used to represent communicative acts, while a combination of net places and color tokens describe the current state of represented conversation. Here, the individual CP-nets, representing separately each conversing agent, are connected using intermediate collective places.

[Mazouzi *et al.*, 2002] presented a similar concept. However, in their representation, places and transitions have slightly different semantics: places represent interaction states and transitions describe actions performed on the corresponding communicative acts (e.g. send, receive and process). The individual CP-nets are connected using synchronization places which connect a send transition of one CP-net to a receive transition of another CP-net. In addition, the authors introduce a recursive CP-net model for representing complex multi-agent interactions. The authors demonstrate its use for recursive modelling of complex communicative acts and interaction sub-protocols.

Finally, investigations by [Nowostawski *et al.*, 2001, Poutakidis *et al.*, 2002] model the entire conversation using a single Petri net corresponding to all conversing agents. In [Nowostawski *et al.*, 2001], places represent message containers, containing communicative acts denoted as color tokens, while transitions describe message events. On the other hand, [Poutakidis *et al.*, 2002] use places to represent both messages and interaction states, whereas transitions are used to model the transmission of the corresponding communicative acts.

As shown in this section, different Petri net representations have been proposed to model multi-agent conversations. However, their relative strength and weaknesses have not been analyzed. In our work, we provide a comprehensive analysis of these previous investigations addressing their scalability and appropriateness for overhearing (see Part I, Chapter 3).

We propose a *scalable* Petri net representation for modelling multi-agent conversation protocols. This representation is particularly suited for monitoring via overhearing. Using the proposed Petri net representation, we cover a variety of conversation features not covered by previous investigations. These features include representation of a full set of FIPA interaction building blocks, communicative act attributes (such as message guards, sequence expressions, etc.), compact modelling of concurrent conversations, nested and interleaved interactions, and temporal aspects. This work is presented throughout Part I of this thesis.

2.4 Conversation Recognition

Conversation recognition, as a concept, is inspired by existing investigations on plan recognition [Kautz and Allen, 1986, Charniak and Goldman, 1993, Carrbery, 2001, Kaminka *et al.*, 2002, Geib and Harp, 2004, Avrahami and Kaminka, 2005]. In plan recognition, agents model other agents unobservable plans using their observable actions. Similarly, conversation recognition is used to recognize conversations between agents based on their overheard communications.

Although certain similarity exists between the two approaches, plan recognition and conversation recognition differ significantly. The main focus of plan recognition is the interactions of an agent and its surrounding environment, whereas conversation recognition focuses on interactions between agents. Furthermore, plan recognition uses the observable actions of an agent to determine its state within the monitored settings, while conversation recognition considers the multi-agent state of all agents participating in the monitored conversation.

Conversation recognition is a key step in overhearing. It is a preliminary step to all possible inference based on overhearing (see overhearing applications in Section 2.1). Unfortunately, it is often ignored by previous investigations on overhearing assuming that conversations can easily be recognized based on their intercepted messages. In addition, most previous works assume that conversations are overheard just as they are carried out between the conversing agents, i.e. all exchanged messages are accessible to the overhearing agent. However, this assumption can be challenged in real-world settings where a difference may exist between the actual and the overheard conversation. For instance, some messages may not reach the overhearing agent due to noise, a temporary malfunction, etc. Furthermore, even investigations addressing such lossy cases [Kaminka *et al.*, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005] assume conversation recognition to be possible, but do not provide any algorithms for handling these non-trivial cases.

Our work was the first to tackle conversation recognition in context of overhearing. This work is presented in Part II of this thesis. We first propose a formal model for the general problem of overhearing. Then, using this model, we formalize the process of conversation recognition providing algorithms for handling both lossless and lossy cases of overhearing.

However, our work on conversation recognition constituted only the first steps in that direction. Conversation recognition algorithms, presented in Chapter 7, addressed conversations where each agent is performing a certain role throughout the conversation. This is not the case in multi-party communications recently gaining interest in multi-agent systems [Kumar *et al.*, 2000, Traum and Rickel, 2002, Dignum and Vreeswijk, 2003]. In such communications, every agent, participating in a conversation, can implement any given role at any given time. Thus, conversation recognition algorithms should be adapted for handling multi-party dialogues. Later work by [Fan and Yen, 2005], based on our investigation, studies conversation recognition for multi-party communications. However, this still remains a challenging problem open for future research.

2.5 Selective Overhearing

All existing applications using overhearing assume the ability to overhear *all* inter-agent communications. This assumption is often challenged in real-world settings, and in particular, in *large-scale* MAS. Instead, the overhearing agent has limited resources, and may only overhear a subset of conversations carried out in the monitored settings. Consequently, the overhearer must be selective in choosing which conversations must be overheard.

Few previous investigations have touched overhearing where some conversations may not be accessible. However, these focused on lossy cases, where due to noise or some other factors, some messages may not reach the overhearing agent. The work by [Kaminka *et al.*, 2002] used plan recognition in overhearing a distributed team of agents. They evaluated the use of their system in lossy settings, and showed that the performance of the overhearing agent drops when messages are lost. The investigations by [Rossi and Busetta, 2004, Rossi and Busetta, 2005] applied overhearing to monitor changes in MAS settings caused by transition from one state to another. They mention that lost messages can cause inconsistencies. Also, our work on conversation recognition (see Part II of the thesis) presents algorithms for conversation recognition, that are robust to lost messages.

In contrast to all of these, we deal here with the case where complete conversations are unknown to the overhearing agent, rather than messages. Moreover, in selective overhearing, the number of conversations unknown to the overhearer is typically much greater than the number of conversations overheard, while in lossy settings, the reverse is typically true.

Our work in Part III of the thesis takes the first steps in this direction. Here, we address selective overhearing based on organizational knowledge of the monitored settings. Organizations have long been studied in the various disciplines of social science [Selznick, 1948, Morgenstern, 1951, Friedell, 1967]. In the multi-agent community, agent-organizations have only recently regained interest [Corkill and Lander, 1998, Dignum, 2003, Horling and Lesser, 2004, Grossi *et al.*, 2005].

In our study, we focus on selective overhearing of hierarchical organizations. Hierarchical organizations are common both in the real-world settings (e.g., many corporates) and in the implementations of multi-agent systems. In the latter settings, hierarchical structures are mainly used for decomposition of complex tasks (e.g., in [So and Durfee, 1996, Yadgar *et al.*, 2003]).

Overhearing hierarchical organizations, we take into consideration both their specific characteristics and the limitation of selectivity implied by overhearing large-scale settings. Thus, we first propose a model of communications in hierarchical organizations (see Part III, Chapter 8). This model is based on investigations of hierarchical communications taken from the different fields of social science [Dewan *et al.*, 1997, Best *et al.*, 2003, Gannon and Newman, 2001, Jensen, 2003, Friebel and Raith, 2004].

Moreover, selective overhearing assumes that conversations may differ in the value of information derived from overhearing them. Otherwise, a random decision, on choosing which conversations to overhear, would be sufficient. To express the possible differences between the overhearing value of conversations, we utilize the concept of *value of information* commonly used in information theory [Packel *et al.*, 1992, Traub and Werschulz, 1998]. Recently, this concept has also been applied in various investigations on distributed artificial intelligence and multi-agent systems. For instance, [Horty and Pollack, 2001] used it as a basis for decision making in context of plans, while [Wilkins *et al.*, 2003] applied value of information to decide on importance of alerts.

Our work in Part III of the thesis utilizes the concept of value of information in context of selective overhearing: It describes the value of information derived from overhearing conversations carried out in monitored organization. Based on this value, the overhearing agent chooses which conversations it would overhear, and which it would not. In hierarchical organizations, which are the focus of our research in this thesis, conversations in different levels of organization vary both in their quantity [Jensen, 2003] and their quality (i.e., in terms of their value) [Best *et al.*, 2003, Gannon and Newman, 2001]. Taking these differences into consideration, we study how to effectively overhear hierarchical organizations under the limitation of selectivity.

Based on the proposed model, we simulate communications in hierarchical organizations and empirically study different centralized and distributed overhearing policies. In doing so, we tackle another problematic assumption made by previous investigations on overhearing assuming the use of a single overhearing agent. This assumption can be challenged in open distributed multi-agent settings. In such settings, monitored agents are often geographically distributed. Therefore, unless some sort of report-based policy is enforced, a single overhearer will be able to overhear only a small subset of potential targets.

Multiple overhearing agents can be deployed to increase the coverage of overheard targets. Few previous investigations addressed the case of multiple overhearing agents [Novick and Ward, 1993, Legras, 2002]. However, these works assume overhearing agents to be non-collaborative entities, committing overhearing out of their own self interest. A non-collaborative overhearer is equivalent to a single overhearer working on its own. Accordingly, these multiple non-collaborative overhearers are still facing the problems of a single centrally-located overhearing agent.

In contrast, we examine teams of overhearing agents collaboratively overhearing multi-agent settings. Indeed, centralized overhearing policies, examined initially in Chapter 9, can be seen as equivalent to a single overhearing agent. However, the later study of distributed overhearing policies in Chapter 10 tackles the assumption of a single centrally-located overhearer.

Part I

Representing Conversations for Scalable Overhearing

Lately, Petri net formalism has been gaining acceptance in multi-agent community. In particular, a variety of Petri net approaches have been proposed to represent multi-agent conversations as shown in Section 2.3. However, addressing those representations, some key questions remain open. First, while radically different approaches to representation using Petri nets have been proposed, their relative strengths and weaknesses have not been investigated. Second, many investigations have only addressed restricted subsets of the features needed in representing complex conversations such as those standardized by FIPA (*Foundation for Intelligent Physical Agents*) [FIPA site, 2005]. Finally, no procedures have been proposed for translating human-readable AUML protocol descriptions into the corresponding machine-readable Petri nets.

This part of the thesis addresses these open challenges in the context of scalable *overhearing*. Here, an overhearing agent passively tracks many concurrent conversations involving multiple participants, based solely on their exchanged messages, while not being a participant in any of the overheard conversations itself [Aiello *et al.*, 2001, Busetta *et al.*, 2001, Kaminka *et al.*, 2002, Legras, 2002]. For instance, an overhearing agent may monitor the conversation of a contractor agent engaged in multiple contract-net protocols with different bidders and bid callers, in order to monitor for any failures.

We begin with an analysis of Petri net representations, with respect to scalability and overhearing (Chapter 3). We classify previous representation choices along two dimensions affecting scalability: (i) the technique used to represent multiple concurrent conversations; and (ii) the choice of representing either individual or joint conversation states. While the run-time complexity of monitoring conversations using different approaches is the same, choices along these two dimensions have significantly different space requirements, and thus some choices are more scalable (in the number of conversations) than others. We also argue that representations suitable for overhearing require the use of explicit message places, though only a subset of previously-explored techniques utilized those.

Building on the insights gained, we propose a novel representation that uses Colored Petri nets (CP-nets) in which places explicitly denote messages, and valid joint conversation states (Chapter 4). This representation is particularly suited for overhearing as the number of conversations is scaled-up. We show how this representation can be used to represent essentially all features of FIPA AUML conversation standards, including simple and complex interaction building blocks, communicative act attributes such as message guards and cardinalities, nesting, and temporal aspects such as deadlines and duration.

To realize the advantages of machine-readable representations, such as for debugging [Poutakidis *et al.*, 2002], existing human-readable protocol descriptions must be converted to their corresponding Petri net representations. As a final contribution of our work in this part of the thesis, we provide a skeleton semi-automated procedure for converting FIPA conversation protocols in AUML to Petri nets, and demonstrate its use on a complex FIPA protocol (Chapter 5). While this procedure is not fully automated, it takes a first step towards addressing this open challenge.

Chapter 3

Classification & Analysis of Conversation Representations

Overhearing involves monitoring conversations as they progress, by tracking messages that are exchanged between participants. We are interested in representations that can facilitate *scalable* overhearing, tracking many concurrent conversations, between many agents. We focus on open settings, where the complex internal state and control logic of agents is not known in advance, and therefore exclude discussions of Petri net representations which explicitly model agent internals (e.g. the works by [Moldt and Wienberg, 1997, Xu and Shatz, 2001]). Instead, we treat agents as black boxes, and consider representations that commit only to the agent's conversation state (i.e., its role and progress in the conversation).

The suitability of a representation for scalable overhearing is affected by several facets. First, since overhearing is based on tracking messages, the representation must be able to explicitly represent the passing of a message (communicative act) from one agent to another (Section 3.1). Second, the representation must facilitate tracking of multiple concurrent conversations. While the tracking runtime is bounded from below by the number of messages (since in any case, all messages are overheard and processed), space requirements may differ significantly (see Sections 3.2–3.3).

3.1 Message monitoring vs. state monitoring

We distinguish two settings for tracking the progress of conversations, depending on the information available to the tracking agent. In the first type of setting, which we refer to as *state monitoring*, the tracking agent has access to the *internal* state of the conversation in one or more of the participants, but not necessarily to the messages being exchanged. The other settings involves *message monitoring*, where the tracking agent has access only to the messages being exchanged (which are *externally observable*), but cannot directly observe the internal state of the conversation in each participant. Overhearing is a form of message monitoring.

Representations that support state monitoring use places to denote the conversation states of the participants. Tokens placed in these places (the net marking) denote the current state. The sending or receiving of a message by a participant is not explicitly represented, and is instead implied by moving tokens (through transition firings) to the new state places. Thus, such a representation essentially assumes that the conversation state of participants is directly observable by the monitoring agent. Previous work utilizing state monitoring includes investigations by [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Mazouzi *et al.*, 2002, Ramos *et al.*, 2002].

The representation we present in this work is intended for overhearing tasks, and cannot assume that the conversation states of overheard agents are observable. Instead, it must support message monitoring, where in addition to using tokens in state places (to denote current conversation state), the representation uses *message places*, where tokens are placed when a corresponding message is overheard. A conversation-state place and a message place are connected via a transition to a state place denoting the new conversation state. Tokens placed in these originating places–indicating a message was received at an appropriate conversation state–will cause the transition to fire, and for the tokens to be placed in the new conversation state place. Thus the new conversation state is inferred from "observing" a message. Previous investigations, that have used explicit message places, include work by [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Nowostawski *et al.*, 2001, Purvis *et al.*, 2002, Cranefield *et al.*, 2002, Poutakidis *et al.*, 2002][1]. These are discussed in depth below.

[1][Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000] present examples of both state- and message-monitoring representations.

3.2 Representing a Single Conversation

Two representation variants are popular within those that utilize conversation places (in addition to message places): *Individual state representations* use separate places and tokens for the state of each participant (each role). Thus, the overall state of the conversation is represented by different tokens marking multiple places. *Joint state representations* use a single place for each joint conversation state of all participants. The placement of a token within such a place represents the overhearing agent's belief that the participants are in the appropriate joint state.

Most previous representations use individual states. In these, different markings distinguish a conversation state where one agent has sent a message, from a state where the other agent received it. The net for each conversation role is essentially built separately, and is either merged with the other nets [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Mazouzi *et al.*, 2002], or connected to them via fusion places, special I/O places and transitions or similar means [Purvis *et al.*, 2002, Ramos *et al.*, 2002, Cranefield *et al.*, 2002].

In principle, individual-state representations require two places in each role, for every message. For a given message, there would be two individual places for the sender (before sending and after sending), and similarly two more *for each receiver* (before receiving and after receiving). All possible conversation states–valid or not–can be represented. For a single message and two roles, there are two places for each role (four places total), and four possible conversation states: message sent and received, sent and not received, not sent but incorrectly believed to have been received, not sent and not received. These states can be represented by different markings. For instance, a conversation state where the message has been sent but not received is denoted by a token in the *'after-sending'* place of the *sender* and another token in the *'before-receiving'* place of the *receiver*. This is summarized in the following proposition:

Proposition 1 *Given a conversation with R roles and a total of M possible messages, an individual state representation has space complexity of $O(MR)$.*

While the representations above all represent each role's conversation state separately, many applications of overhearing only require representation of valid conversation states (message not sent and not received, or sent and received). Indeed, specifications for interaction protocols often assume the use of underlying synchronization protocols to guarantee delivery of messages [Paurobally and Cunningham, 2003, Paurobally *et al.*, 2003]. Under such an assumption, for every message, there are only two joint states regardless of the number of roles. For example, for a single message and three roles–a sender and two receivers, there are two places and two possible markings: A token in a *before sending/receiving* place represents a conversation state where the message has not yet been sent by the *sender* (and the two *receivers* are waiting for it), while a token in a *after sending/receiving* place denotes that the message has been sent and received by both *receivers*.

[Nowostawski *et al.*, 2001] utilize CP-nets where places denote joint conversation states. They also utilize places representing communicative acts. The work by [Poutakidis *et al.*, 2002] proposed a representation based on Place-Transition nets (PT-nets)–a more restricted representation of Petri nets that has no color. They presented several interaction building blocks, which could then fit together to model additional conversation protocols. In general the following proposition holds with respect to such representations:

Proposition 2 *Given a conversation with R roles and a total of M possible messages, a joint state representation that represents only legal states has space complexity of $O(M)$.*

The condition of representing only *valid* states is critical to the complexity analysis. If all joint conversation states–valid and invalid–are to be represented, the space complexity would be $O(M^R)$. In such a case, an individual-state representation would have an advantage. This would be the case, for instance, if we do not assume the use of synchronization protocols, e.g., where the overhearing agent may wish to track the exact system state even while a message is underway (i.e., sent and not yet received).

3.3 Representing Multiple Concurrent Conversations

Propositions 1 and 2 above address the space complexity of representing a single conversation. However, in large scale systems an overhearing agent may be required to monitor multiple conversations in parallel. For instance, an overhearing agent may be monitoring a middle agent that is carrying multiple parallel instances of a single interaction protocol with multiple partners, e.g., brokering [FIPA Specifications, 2005a].

Some previous investigations [Nowostawski *et al.*, 2001, Poutakidis *et al.*, 2002] propose to duplicate the appropriate Petri net representation for each monitored conversation. In this approach, every conversation is tracked by a separate Petri-net, and thus the number of Petri nets (and their associated tokens) grows with the number of conversations (Proposition 3). For instance, [Nowostawski *et al.*, 2001] shows an example where a contract-net protocol is carried out with three different contractors, using three duplicate CP-nets. This is captured in the following proposition:

Proposition 3 *A representation that creates multiple instances of a conversation Petri net to represent C conversations, requires $O(C)$ net structures, and $O(C)$ bits for all tokens.*

Other investigations take a different approach, in which a single CP-net structure is used to monitor all conversations of the same protocol. The tokens (on the same CP-net structure) associated with different conversations are distinguished by their token color [Cost, 1999, Cost *et al.*, 1999, Cost *et al.*, 2000, Lin *et al.*, 2000, Purvis *et al.*, 2002, Ramos *et al.*, 2002, Mazouzi *et al.*, 2002, Cranefield *et al.*, 2002]. For example, by assigning each token a color of the tuple type $\langle sender, receiver \rangle$, an agent can differentiate multiple tokens in the same place and thus track conversations of different pairs of agents[2]. Color tokens use multiple bits per token; up to $\log C$ bits are required to differentiate C conversations. Therefore, the number of bits required to track C conversations using C tokens is $C \log C$. This leads to the following proposition.

Proposition 4 *A representation that uses color tokens to represent C multiple instances of a conversation, requires $O(1)$ net structures, and $O(C \log C)$ bits for all tokens.*

Due to the constants involved, the space requirements of Proposition 3 are in practice much more expensive than those of Proposition 4. Proposition 3 refers to the creation of $O(C)$ Petri networks, each with duplicated place and transition data structures. In contrast, Proposition 4 refers to bits required for representing C color tokens on a single CP net. Moreover, in most practical settings, a sufficiently large constant bound on the number of conversations may be found, which will essentially reduce the $O(\log C)$ factor to $O(1)$.

[2]See Section 4.2 to distinguish between different conversations by the same agents.

	Representing Multiple Conversations (of Same Protocol)	
	Multiple CP- or PT-nets (Proposition 3)	**Using color tokens, single CP-net** (Proposition 4)
Individual States (Proposition 1)	**Space:** $O(MRC)$	**Space:** $O(MR + C \log C)$ [Cost, 1999, Cost *et al.*, 1999], [Cost *et al.*, 2000, Lin *et al.*, 2000], [Cranefield *et al.*, 2002], [Ramos *et al.*, 2002] [Purvis *et al.*, 2002] [Mazouzi *et al.*, 2002]
Joint States (Proposition 2)	**Space:** $O(MC)$ [Nowostawski *et al.*, 2001], [Poutakidis *et al.*, 2002]	**Space:** $O(M + C \log C)$ Our work

Table 3.1: Scalability of different representations

Based on Propositions 1–4, it is possible to make concrete predictions as to the scalability of different approaches with respect to the number of conversations, roles. Table 3.1 shows the space complexity of different approaches when modelling C conversations of the same protocol, each with a maximum of R roles, and M messages, under the assumption of underlying synchronization protocols. The table also cites relevant previous work.

Building on the insights gained from Table 3.1, we propose a representation using CP-nets where places explicitly represent joint conversation states (corresponding to the lower-right cell in Table 3.1), and tokens color is used to distinguish concurrent conversations (as in the upper-right cell in Table 3.1). As such, it is related to the works that have these features, but as the table demonstrates, is a novel synthesis.

Our representation uses similar structures to those found in the investigations by [Nowostawski *et al.*, 2001, Poutakidis *et al.*, 2002]. However, in contrast to these previous investigations, we rely on token color in CP-nets to model concurrent conversations, with space complexity $O(M + C \log C)$. We also show (Chapter 4) how it can be used to cover a variety of conversation features not covered by these investigations. These features include representation of a full set of FIPA interaction building blocks, communicative act attributes (such as message guards, sequence expressions, etc.), compact modelling of concurrent conversations, nested and interleaved interactions, and temporal aspects.

Chapter 4

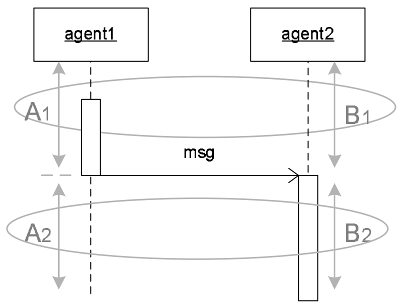
Representing Various Aspects of Conversations

Chapter 3 presented the motivation for our work analyzing the previous Petri net representations of multi-agent conversations in terms of their scalability and appropriateness for overhearing. In this chapter, we present the proposed representation, based on the analysis found in the previous chapter, addressing all FIPA conversation features including basic interaction building blocks (Section 4.1), message attributes (Section 4.2), nested & interleaved interactions (Section 4.3), and temporal aspects (Section 4.4). To conclude this chapter, based on these features, Section 4.5 presents a Petri net of a complex conversation protocol, which integrates many of the features of the developed representation technique.

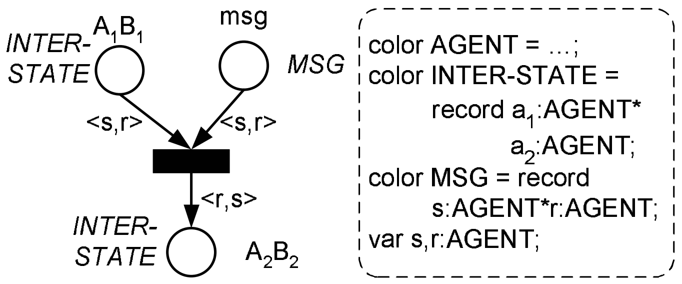
4.1 Basic Conversation Building Blocks

This section introduces the fundamentals of our representation, and demonstrates how various simple and complex AUML interaction messages, used in FIPA conversation standards [FIPA Specifications, 2005c], can be implemented using the proposed CP-net representation. We begin with a simple conversation, shown in Figure 4.1-a using an AUML protocol diagram. Here, $agent_1$ sends an asynchronous message msg to $agent_2$.

To represent agent conversation protocols, we define two types of places, corresponding to messages and conversation states. The first type of net places, called *message places*, is used to describe conversation communicative acts. Tokens placed in message places indicate that the associated communicative act has been over-



(a) AUML representation



(b) CP-net representation

Figure 4.1: Asynchronous message interaction.

heard. The second type of net places, *agent places*, is associated with the valid *joint* conversation states of the interacting agents. Tokens placed in agent places indicate the current joint state of the conversation within the interaction protocol.

Transitions represent the transmission and receipt of communicative acts between agents. Assuming underlying synchronization protocols, a transition always originates within a joint-state place and a message place, and targets a joint conversation state (more than one is possible–see below). Normally, the current conversation state is known (marked with a token), and must wait the overhearing of the matching message (denoted with a token at the connected message place). When this token is marked, the transition fires, automatically marking the new conversation state.

Figure 4.1-b presents CP-net representation of the earlier example of Figure 4.1-a. The CP-net in Figure 4.1-b has three places and one transition connecting them. The A_1B_1 and the A_2B_2 places are agent places, while the *msg* place is a message place. The A and B capital letters are used to denote the $agent_1$ and the $agent_2$ individual interaction states respectively (we have indicated the individual and the joint interaction states over the AUML diagram in Figure 4.1-a, but omit these annotations in later figures). Thus, the A_1B_1 place indicates a joint interaction state where $agent_1$ is ready to send the *msg* communicative act to $agent_2$ (A_1) and $agent_2$ is waiting to receive the corresponding message (B_1). The *msg* message place corresponds to the *msg* communicative act sent between the two agents. Thus, the transmission of the *msg* communicative act causes the agents to transition to the A_2B_2 place. This place corresponds to the joint interaction state in which $agent_1$ has already sent the *msg* communicative act to $agent_2$ (A_2) and $agent_2$ has received it (B_2).

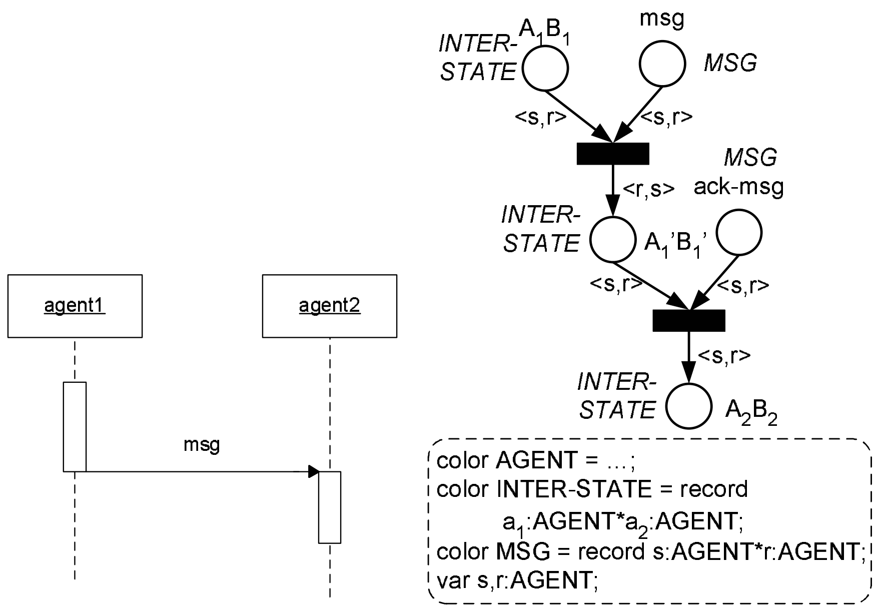
The CP-net implementation in Figure 4.1-b also introduces the use of token colors to represent additional information about interaction states and communicative acts. The token color sets are defined in the net declaration, i.e. the dashed box in Figure 4.1-b. The syntax follows the standard CPN ML notation [Wikstrom, 1987, Milner *et al.*, 1990, Jensen, 1997a]. The *AGENT* color identifies the agents participating in the interaction, and is used to construct the two compound color sets.

The *INTER-STATE* color set is associated with agent places, and represents agents in the appropriate joint interaction states. It is a record $\langle a_1, a_2 \rangle$, where a_1 and a_2 are *AGENT* color elements distinguishing the interacting agents. We apply the *INTER-STATE* color set to model multiple concurrent conversations using the same CP-net. The second color set is *MSG*, describing interaction communicative acts and associated with message places. The *MSG* color token is a record $\langle a_s, a_r \rangle$, where a_s and a_r correspond to the sender and the receiver agents of the associated communicative act. In both cases, additional elements, such as conversation identification, may be used. See Section 4.2 for additional details.

In Figure 4.1-b, the A_1B_1 and the A_2B_2 places are associated with the *INTER-STATE* color set, while the *msg* place is associated with the *MSG* color set. The place color set is written in italic capital letters next to the corresponding place. Furthermore, we use the s and r *AGENT* color type variables to denote the net arc expressions. Thus, given that the output arc expression of both the A_1B_1 and the *msg* places is $< s, r >$, the s and r elements of the agent place token must correspond to the s and r elements of the message place token. Consequently, the net transition occurs if and only if the agents of the message correspond to the interacting agents. The A_2B_2 place input arc expression is $< r, s >$ following the underlying intuition that $agent_2$ is going to send the next interaction communicative act.

Figure 4.2-a shows an AUML representation of another interaction building block, synchronous message passing, denoted by the filled solid arrowhead. Here, the *msg* communicative act is sent synchronously from $agent_1$ to $agent_2$, meaning that an acknowledgement on *msg* communicative act must always be received by $agent_1$ before the interaction may proceed.

The corresponding CP-net representation is shown in Figure 4.2-b. The interaction starts in the A_1B_1 place and terminates in the A_2B_2 place. The A_1B_1 place represents a joint interaction state where $agent_1$ is ready to send the *msg* communicative act to $agent_2$ (A_1) and $agent_2$ is waiting to receive the corresponding



```
color AGENT = ...;
color INTER-STATE = record
        a1:AGENT*a2:AGENT;
color MSG = record s:AGENT*r:AGENT;
var s,r:AGENT;
```

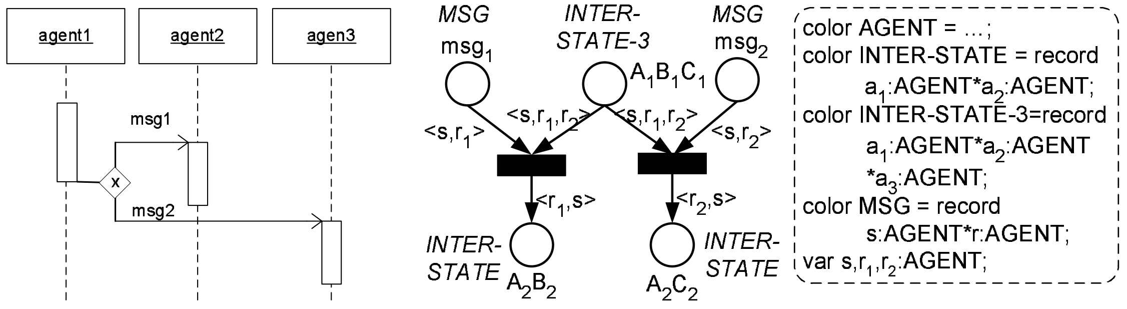
(a) AUML representation (b) CP-net representation

Figure 4.2: Synchronous message interaction.

message (B_1). The A_2B_2 place denotes a joint interaction state, in which $agent_1$ has already sent the *msg* communicative act to $agent_2$ (A_2) and $agent_2$ has received it (B_2). However, since the CP-net diagram represents synchronous message passing, the *msg* communicative act transmission cannot cause the agents to transition directly from the A_1B_1 place to the A_2B_2 place. We therefore define an intermediate $A_1'B_1'$ agent place. This place represents a joint interaction state where $agent_2$ has received the *msg* communicative act and is ready to send an acknowledgement on it (B_1'), while $agent_1$ is waiting for that acknowledgement (A_1'). Taken together, the *msg* communicative act causes the agents to transition from the A_1B_1 place to the $A_1'B_1'$ place, while the acknowledgement on the *msg* message causes the agents to transition from the $A_1'B_1'$ place to the A_2B_2 place.

Nevertheless, transitions in a typical multi-agent interaction protocols are often not as simple as those described above. Thus, the remaining part of this section is dedicated to the complex interaction transitions that may appear in a standard multi-agent conversation.

We begin with the XOR-decision interaction. The AUML representation to this building block is shown in Figure 4.3-a. The sender agent $agent_1$ can either send message msg_1 to $agent_2$ or message msg_2 to $agent_3$, but it can not send both msg_1 and msg_2. The non-filled diamond with an 'x' inside is the AUML notation for this constraint.



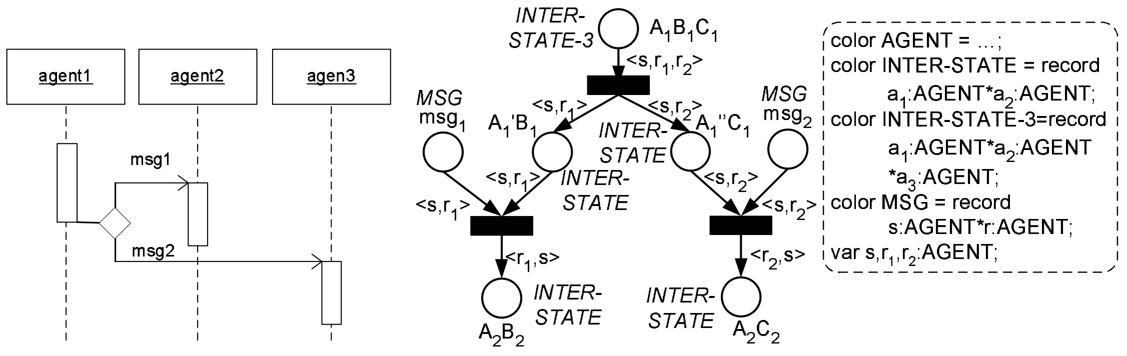
(a) AUML representation (b) CP-net representation

Figure 4.3: XOR-decision messages interaction.

Figure 4.3-b shows the corresponding CP-net. Again, the A, B and C capital letters are used to denote the interaction states of $agent_1$, $agent_2$ and $agent_3$, respectively. The interaction starts from the $A_1B_1C_1$ place and terminates either in the A_2B_2 place or in the A_2C_2 place. The $A_1B_1C_1$ place represents a joint interaction state where $agent_1$ is ready to send either the msg_1 communicative act to $agent_2$ or the msg_2 communicative act to $agent_3$ (A_1); and $agent_2$ and $agent_3$ are waiting to receive the corresponding msg_1/msg_2 message (B_1/C_1). To represent the $A_1B_1C_1$ place color set, we extend the *INTER-STATE* color set to denote a joint interaction state of three interacting agents, i.e. using the *INTER-STATE-3* color set. The msg_1 communicative act causes the agents to transition to A_2B_2 place. The A_2B_2 place represents a joint interaction state where $agent_1$ has sent the msg_1 message (A_2), and $agent_2$ has received it (B_2). Similarly, the msg_2 communicative act causes agents $agent_1$ and $agent_3$ to transition to A_2C_2 place. Exclusiveness is achieved since the single agent token in $A_1B_1C_1$ place can be used either for activating the $A_1B_1C_1 \rightarrow A_2B_2$ transition or for activating the $A_1B_1C_1 \rightarrow A_2C_2$ transition, but not both.

A similar complex interaction is the OR-parallel messages interaction. Its AUML representation is presented in Figure 4.4-a. The sender agent, $agent_1$, can send message msg_1 to $agent_2$ or message msg_2 to $agent_3$, or both. The non-filled diamond is the AUML notation for this constraint.

Figure 4.4-b shows the CP-net representation of the OR-parallel interaction. The interaction starts from the $A_1B_1C_1$ place but it can be terminated in the A_2B_2 place, or in the A_2C_2 place, or in both. To represent this inclusiveness of the interaction protocol, we define two intermediate places, the $A_1'B_1$ place and the $A_1''C_1$ place. The $A_1'B_1$ place represents a joint interaction state where $agent_1$ is ready to send



```
color AGENT = ...;
color INTER-STATE = record
      a1:AGENT*a2:AGENT;
color INTER-STATE-3=record
      a1:AGENT*a2:AGENT
      *a3:AGENT;
color MSG = record
      s:AGENT*r:AGENT;
var s,r1,r2:AGENT;
```

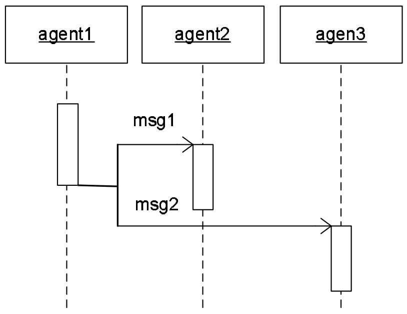
(a) AUML representation (b) CP-net representation

Figure 4.4: OR-parallel messages interaction.

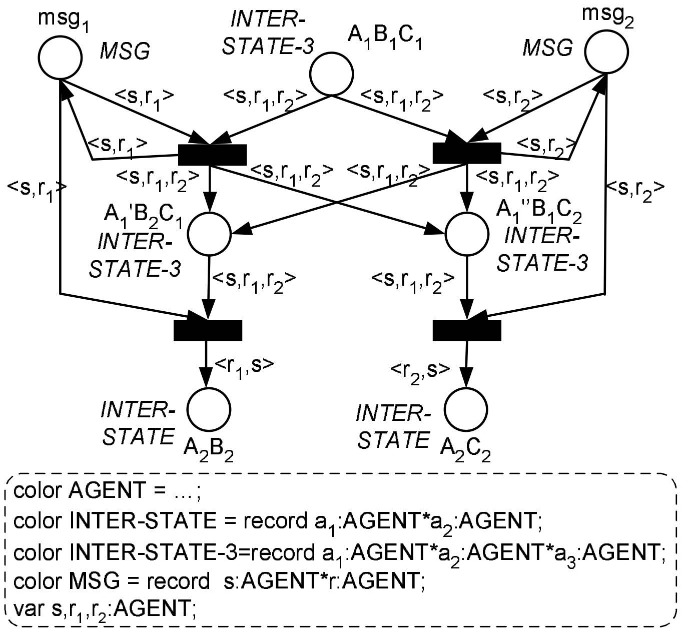
the msg_1 communicative act to $agent_2$ (A_1') and $agent_2$ is waiting to receive the message (B_1). The $A_1''C_1$ place has similar meaning, but with respect to $agent_3$. As normally done in Petri nets, the transition connecting the $A_1B_1C_1$ place to the intermediate places duplicates any single token in $A_1B_1C_1$ place into two tokens going into the $A_1'B_1$ and the $A_1''C_1$ places. Consequently, the two parts of the OR-parallel interaction can be independently executed.

Next, the AND-parallel messages interaction is presented. Its AUML representation is shown in Figure 4.5-a. Here, the sender $agent_1$ sends both the msg_1 message to $agent_2$ and the msg_2 message to $agent_3$. However, the order of the two communicative acts is unconstrained.

The representation of AND-parallel in our CP-net representation is shown in Figure 4.5-b. The $A_1B_1C_1$, A_2B_2, A_2C_2, msg_1 and msg_2 places are defined similarly to Figures 4.3-b and 4.4-b above. However, we also define two additional intermediate agent places, $A_1'B_2C_1$ and $A_1''B_1C_2$. The $A_1'B_2C_1$ place represents a joint interaction state where $agent_1$ has sent the msg_1 message to $agent_2$ and is ready to send the msg_2 communicative act to $agent_3$ (A_1'), $agent_2$ has received the msg_1 message (B_2) and $agent_3$ is waiting to receive the msg_2 communicative act (C_1). The $A_1''B_1C_2$ place represents a joint interaction state in which $agent_1$ is ready to send the msg_1 message to $agent_2$ and has already sent the msg_2 communicative act to $agent_3$ (A_1''), $agent_2$ is waiting to receive the msg_1 message (B_1) and $agent_3$ has received the msg_2 communicative act (C_2). These places enable $agent_1$ to send both communicative acts concurrently. Four transitions connect the appropriate places respectively. The behavior of the transitions connecting $A_1'B_2C_1 \rightarrow A_2B_2$ and $A_1''B_1C_2 \rightarrow A_2C_2$ is similar to described above. The transitions $A_1B_1C_1 \rightarrow A_1'B_2C_1$ and $A_1B_1C_1 \rightarrow A_1''B_1C_2$



(a) AUML representation



```
color AGENT = ... ;
color INTER-STATE = record a1:AGENT*a2:AGENT;
color INTER-STATE-3=record a1:AGENT*a2:AGENT*a3:AGENT;
color MSG = record  s:AGENT*r:AGENT;
var s,r1,r2:AGENT;
```

(b) CP-net representation

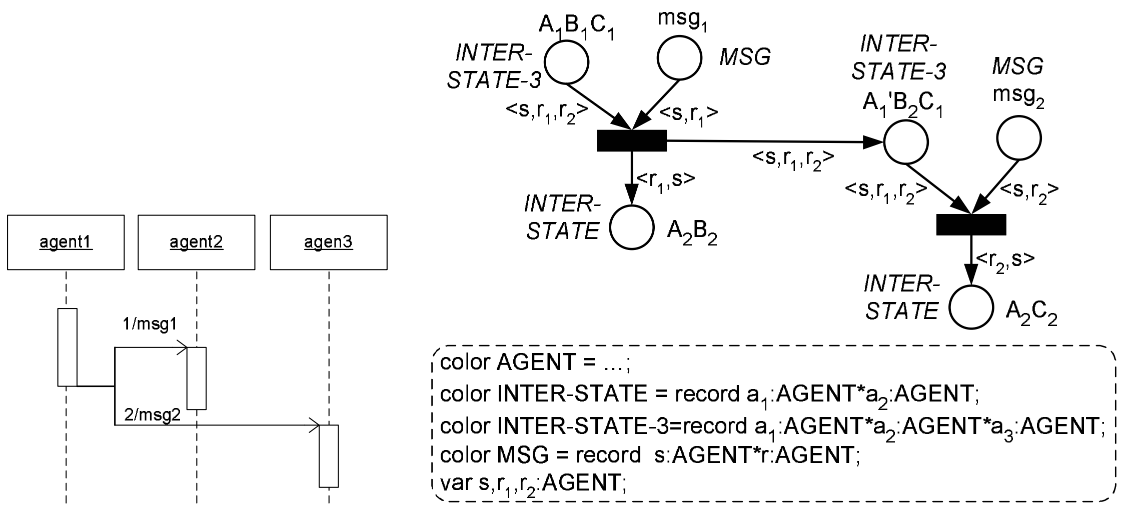
Figure 4.5: AND-parallel messages interaction.

are triggered by receiving messages msg_1 and msg_2, respectively. However, these transitions should not consume the message token since it is used further for triggering transitions $A_1'B_2C_1 \rightarrow A_2B_2$ and $A_1''B_1C_2 \rightarrow A_2C_2$. This is achieved by adding an appropriate message place as an output place of the corresponding transition.

The fourth complex AUML interaction building block, shown in Figure 4.6-a, is the message sequence interaction, which is similar to AND-parallel. However, the message sequence interaction defines explicitly the order between the transmitted messages. Using the $1/msg_1$ and $2/msg_2$ notation, Figure 4.6-a specifies that the msg_1 message should be sent before sending msg_2.

Figure 4.6-b shows the corresponding CP-net representation. The $A_1B_1C_1$, A_2B_2, A_2C_2, msg_1 and msg_2 places are defined as before. However, the CP-net implementation presents an additional intermediate agent place–$A_1'B_2C1$–which is identical to the corresponding intermediate agent place in Figure 4.5-b. $A_1'B_2C_1$ is defined as an output place of the $A_1B_1C_1 \rightarrow A_2B_2$ transition. It thus guarantees that the msg_2 communicative act can be sent (represented by the $A_1'B_2C_1 \rightarrow A_2C_2$ transition) only upon completion of the msg_1 transmission (the $A_1B_1C_1 \rightarrow A_2B_2$ transition).

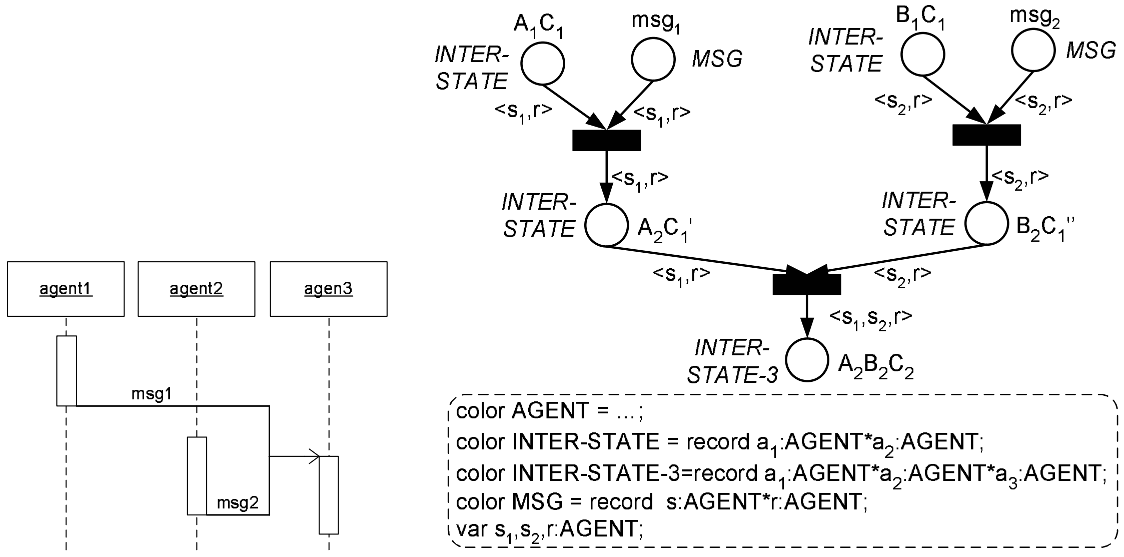
The last interaction we present is the synchronized messages interaction, shown in Figure 4.7-a. Here, $agent_3$ simultaneously receives msg_1 from $agent_1$ and msg_2 from $agent_2$. In AUML, this constraint is annotated by merging the two communicative act arrows into a horizontal bar with a single output arrow.



```
color AGENT = ...;
color INTER-STATE = record a1:AGENT*a2:AGENT;
color INTER-STATE-3=record a1:AGENT*a2:AGENT*a3:AGENT;
color MSG = record  s:AGENT*r:AGENT;
var s,r1,r2:AGENT;
```

(a) AUML representation (b) CP-net representation

Figure 4.6: Sequence messages interaction.



```
color AGENT = ...;
color INTER-STATE = record a1:AGENT*a2:AGENT;
color INTER-STATE-3=record a1:AGENT*a2:AGENT*a3:AGENT;
color MSG = record  s:AGENT*r:AGENT;
var s1,s2,r:AGENT;
```

(a) AUML representation (b) CP-net representation

Figure 4.7: Synchronized messages interaction.

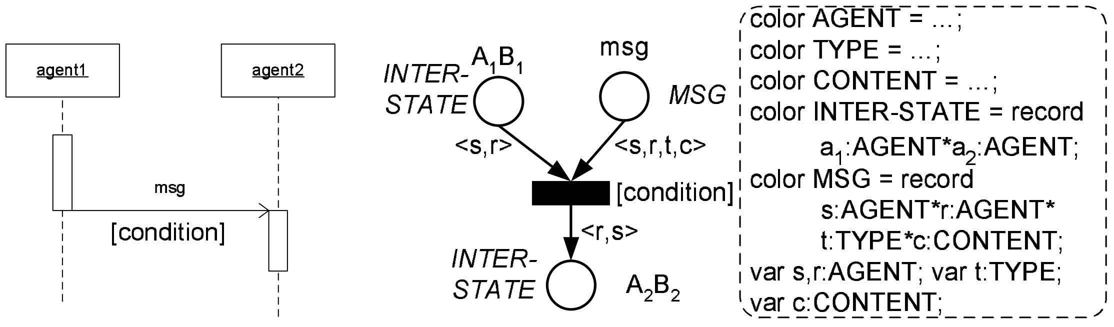
Figure 4.7-b illustrates the CP-net implementation of synchronized messages interaction. As in previous examples, we define the A_1C_1, B_1C_1, msg_1, msg_2 and $A_2B_2C_2$ places. We additionally define two intermediate agent places, A_2C_1' and B_2C_1''. The A_2C_1' place represents a joint interaction state where $agent_1$ has sent msg_1 to $agent_3$ (A_2), and $agent_3$ has received it, however $agent_3$ is also waiting to receive msg_2 (C_1'). The B_2C_1'' place represents a joint interaction state in which

$agent_2$ has sent msg_2 to $agent_3$ (B_2), and $agent_3$ has received it, however $agent_3$ is also waiting to receive msg_1 (C''_1). These places guarantee that the interaction does not transition to the $A_2B_2C_2$ state until both msg_1 and msg_2 have been received by $agent_3$.

4.2 Conversation Attributes

We now extend our representation to allow additional interaction aspects, useful in describing multi-agent conversation protocols. First, we show how to represent interaction message attributes, such as guards, sequence expressions, cardinalities and content [FIPA Specifications, 2005c]. We then explore in depth the representation of multiple concurrent conversations (on the same CP net).

Figure 4.8-a shows a simple agent interaction using an AUML protocol diagram. This interaction is similar to the one presented in Figure 4.1-a in the previous section. However, Figure 4.8-a uses an AUML message guard-condition–marked as [*condition*]–that has the following semantics: the communicative act is sent from $agent_1$ to $agent_2$ if and only if the *condition* is true.



(a) AUML representation (b) CP-net representation

Figure 4.8: Message guard-condition

The guard-condition implementation in our Petri net representation uses transition guards (Figure 4.8-b), a native feature for CP nets. The AUML guard condition is mapped directly to the CP-net transition guard. The CP-net transition guard is indicated on the net inscription next to the corresponding transition using square brackets. The transition guard guarantees that the transition is enabled if and only if the transition guard is true.

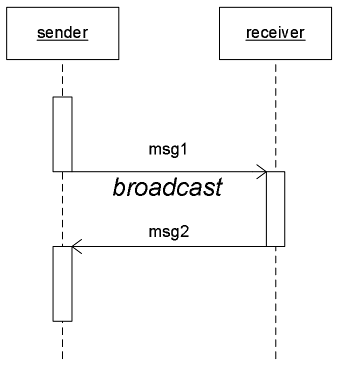
In Figure 4.8-b, we also extend the color of tokens to include information about the communicative act being used and its content. We extend the MSG color set definition to a record $< s, r, t, c >$, where the s and r elements has the same interpretation as in previous section (sender and receiver), and the t and c elements define the message type and content, respectively. The t element is of a new color $TYPE$, which determines communicative act types. The c element is of a new color $CONTENT$, which represents communicative act content and argument list (e.g. reply-to, reply-by and etc).

The addition of new elements also allows for additional potential uses. For instance, to facilitate representation of multiple concurrent conversations between the same agents (s and r), it is possible to add a conversation identification field to both the MSG and *INTER-STATE* colors. For simplicity, we refrain from doing so in the examples in this paper.

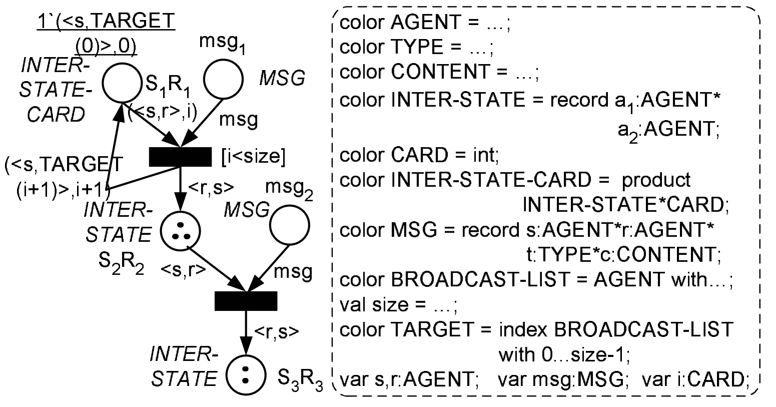
Two additional AUML communicative act attributes that can be modelled in the CP-net representation are message sequence-expression and message cardinality. The sequence-expressions denote a constraint on the message sent from sender agent. There are a number of sequence-expressions defined by FIPA conversation standards [FIPA Specifications, 2005c]: m denotes that the message is sent exactly m times; $n..m$ denotes that the message is sent anywhere from n up to m times; $*$ denotes that the message is sent an arbitrary number of times. An additional important sequence expression is *broadcast*, i.e. message is sent to all other agents.

We now explain the representation of sequence-expressions in CP-nets, using broadcast as an example (Figure 4.9-b). Other sequence expressions are easily derived from this example. We define an *INTER-STATE-CARD* color set. This color set is a tuple $(< a_1, a_2 >, i)$ consisting of two elements. The first tuple element is an *INTER-STATE* color element, which denotes the interacting agents as previously defined. The second tuple element is an integer that counts the number of messages already sent by an agent, i.e. the message cardinality. This element is initially assigned to 0. The *INTER-STATE-CARD* color set is applied to the S_1R_1 place, where the S and R capital letters are used to denote the *sender* and the *receiver* individual interaction states respectively and the S_1R_1 indicates the initial joint interaction state of the interacting agents. The two additional colors, used in Figure 4.9-b, are the *BROADCAST-LIST* and the $TARGET$ colors. The *BROADCAST-LIST* color defines the sender broadcast list of the designated receivers, assuming that the *sender* must have such a list to carry out its role. The $TARGET$ color defines indexes into this broadcast list.

According to the broadcast sequence-expression semantics, the *sender* agent sends the same msg_1 communicative act to all the *receivers* on the *broadcast list*.



(a) AUML representation



```
color AGENT = ...;
color TYPE = ...;
color CONTENT = ...;
color INTER-STATE = record a1:AGENT*
                           a2:AGENT;
color CARD = int;
color INTER-STATE-CARD = product
                  INTER-STATE*CARD;
color MSG = record s:AGENT*r:AGENT*
                  t:TYPE*c:CONTENT;
color BROADCAST-LIST = AGENT with...;
val size = ...;
color TARGET = index BROADCAST-LIST
                  with 0...size-1;
var s,r:AGENT;  var msg:MSG;  var i:CARD;
```

(b) CP-net representation

Figure 4.9: Broadcast sequence expression.

The CP-net introduced in Figure 4.9-b models this behavior.[1] The interaction starts from the S_1R_1 place, representing the joint interaction state where *sender* is ready to send the msg_1 communicative act to *receiver* (S_1) and *receiver* is waiting to receive the corresponding msg_1 message (R_1). The S_1R_1 place initial marking is a single token, set by the initialization expression (underlined, next to the corresponding place). The initialization expression $1`(< s, TARGET(0) >, 0)$–given in standard CPN ML notation–determines that the S_1R_1 place initial marking is a multi-set containing a single token $(< s, TARGET(0) >, 0)$. Thus, the first designated *receiver* is assigned to be the agent with index 0 on the broadcast list, and the message cardinality counter is initiated to 0.

The msg_1 message place initially contains multiple tokens. Each of these tokens represents the msg_1 communicative act addressed to a different designated *receiver* on the *broadcast list*. In Figure 4.9-b, the initialization expression, corresponding to the msg_1 message place, has been omitted. The S_1R_1 place token and the appropriate msg_1 place token together enable the corresponding transition. Consequently, the transition may fire and thus the msg_1 communicative act transmission is simulated.

The msg_1 communicative act is sent incrementally to every designated *receiver* on the *broadcast list*. The incoming arc expression $(< s, r >, i)$ is incremented by the transition to the outgoing $(< s, TARGET(i + 1) >, i + 1)$ arc expression,

[1]We implement broadcast as an iterative procedure sending the corresponding communicative act separately to all designated recipients.

causing the *receiver* agent with index $i + 1$ on the *broadcast list* to be selected. The transition guard constraint $i < size$, i.e. $i < |broadcast\ list|$, ensures that the msg_1 message is sent no more than $|broadcast\ list|$ times. The msg_1 communicative act causes the agents to transition to the S_2R_2 place. This place represents a joint interaction state in which sender has already sent the msg_1 communicative act to *receiver* and is now waiting to receive the msg_2 message (S_2) and *receiver* has received the msg_1 message and is ready to send the msg_2 communicative act to sender (R_2). Finally, the msg_2 message causes the agents to transition to the S_3R_3 place. The S_3R_3 place denotes a joint interaction state where sender has received the msg_2 communicative act from *receiver* and terminated (S_3), while *receiver* has already sent the msg_2 message to *sender* and terminated as well (R_3).

We use Figure 4.9-b to demonstrate the use of token color to represent multiple concurrent conversations using the same CP-net. For instance, let us assume that the *sender* agent is called $agent_1$ and its *broadcast list* contains the following agents: $agent_2$, $agent_3$, $agent_4$, $agent_5$ and $agent_6$. We will also assume that the $agent_1$ has already sent the msg_1 communicative act to all agents on the *broadcast list*. However, it has only received the msg_2 reply message from $agent_3$ and $agent_6$. Thus, the CP-net current marking for the complete interaction protocol is described as follows: the S_2R_2 place is marked by $< agent_2, agent_1 >$, $< agent_4, agent_1 >$, $< agent_5, agent_1 >$, while the S_3R_3 place contains the tokens $< agent_1, agent_3 >$ and $< agent_1, agent_6 >$.

An Example. We now construct a CP-net representation of the *FIPA Query Interaction Protocol* [FIPA Specifications, 2005d], shown in AUML form in Figure 4.10, to demonstrate how the building blocks presented in Sections 4.1 and 4.2 can be put together. In this interaction protocol, the *Initiator* requests the *Participant* to perform an inform action using one of two query communicative acts, *query-if* or *query-ref.* The *Participant* processes the query and makes a decision whether to *accept* or $refuse$ the query request. The *Initiator* may request the *Participant* to respond with either an *accept* or $refuse$ message, and for simplicity, we will assume that this is always the case. In case the *query* request has been accepted, the *Participant* informs the *Initiator* on the query results. If the *Participant* fails, then it communicates a $failure$. In a successful response, the *Participant* replies with one of two versions of inform (*inform-t/f* or *inform-result*) depending on the type of initial query request.

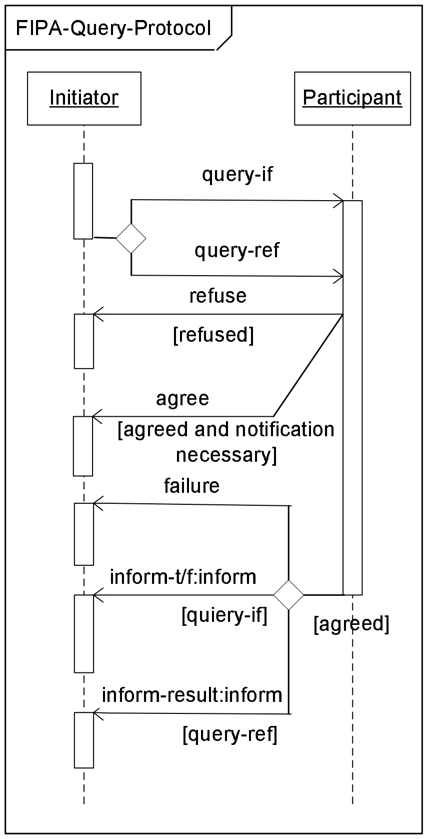


Figure 4.10: FIPA Query Interaction Protocol - AUML representation.

The CP-net representation of the *FIPA Query Interaction Protocol* is presented in Figure 4.11. The interaction starts in the I_1P_1 place (we use the I and the P capital letters to denote the *Initiator* and the *Participant* roles). The I_1P_1 place represents a joint interaction state where (i) the *Initiator* agent is ready to send either the *query-if* communicative act, or the *query-ref* message, to *Participant* (I_1); and (ii) *Participant* is waiting to receive the corresponding message (P_1). The *Initiator* can send either a *query-if* or a *query-ref* communicative act. We assume that these acts belong to the same class, the *query* communicative act class. Thus, we implement both messages using a single *Query* message place, and check the message type using the following transition guard: [*#t msg = query-if* or *#t msg = query-ref*]. The query communicative act causes the interacting agents to transition to the I_2P_2 place. This place represents a joint interaction state in which *Initiator* has sent the *query* communicative act and is waiting to receive a response message (I_2), and *Participant* has received the *query* communicative act and deciding whether to send an *agree* or a *refuse* response message to *Initiator* (P_2). The *refuse* communicative act causes the agents to transition to I_3P_3 place, while the *agree* message causes the agents to transition to I_4P_4 place.

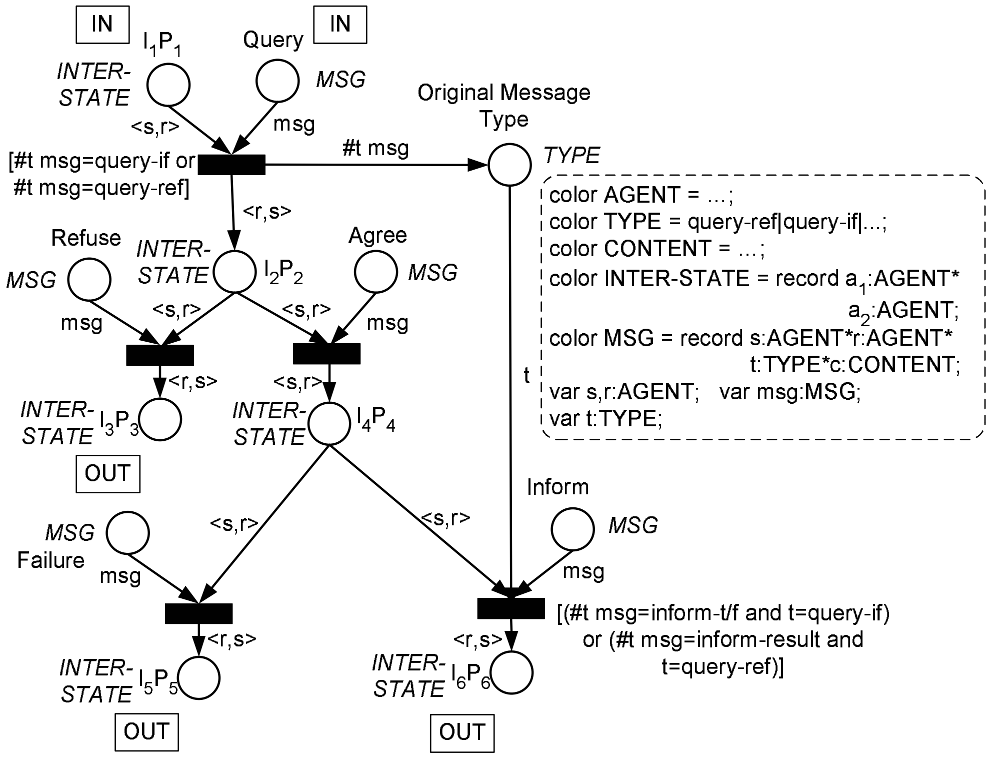


Figure 4.11: FIPA Query Interaction Protocol - CP-net representation.

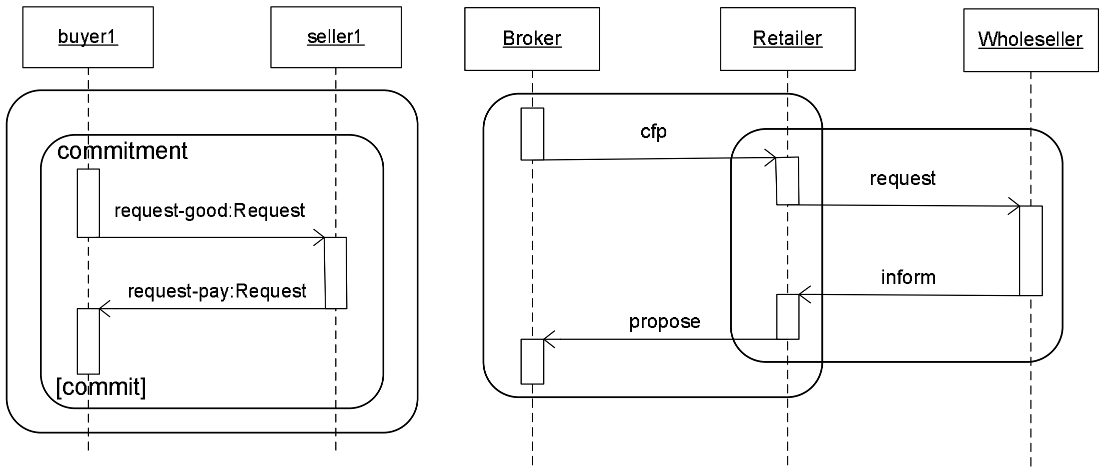
The *Participant* decision on whether to send an *agree* or a *refuse* communicative act is represented using the XOR-decision building block introduced earlier (Figure 4.3-b). The I_3P_3 place represents a joint interaction state where *Initiator* has received a *refuse* communicative act and terminated (I_3) and *Participant* has sent a *refuse* message and terminated as well (P_3). The I_4P_4 place represents a joint interaction state in which *Initiator* has received an *agree* communicative act and is now waiting for further response from *Participant* (I_4) and *Participant* has sent an *agree* message and is now deciding which response to send to *Initiator* (P_4). At this point, the *Participant* agent may send one of the following communicative acts: *inform-t/f*, *inform-result* and *failure*. The choice is represented using another XOR-decision building block, where the *inform-t/f* and *inform-result* communicative acts are represented using a single *Inform* message place. The *failure* communicative act causes transitioning to the I_5P_5 place, while the *inform* message causes a transition to the I_6P_6 place. The I_5P_5 place represents a joint interaction state where *Participant* has sent a *failure* message and terminated (P_5), while *Initiator* has received a *failure* and terminated (I_5). The I_6P_6 place represents a joint interaction state in which *Participant* has sent an *inform* message and terminated (P_6), while *Initiator* has received an *inform* and terminated (I_6).

The implementation of the [*query-if*] and the [*query-ref*] message guard conditions requires a detailed discussion. These are not implemented in a usual manner in view of the fact that they depend on the original request communicative act. Thus, we create a special intermediate place that contains the original message type marked "*Original Message Type*" in the figure. In case an *inform* communicative act is sent, the transition guard verifies that the *inform* message is appropriate to the original *query* type. Thus, an *inform-t/f* communicative act can be sent only if the original query type has been *query-if* and an *inform-result* message can be sent only if the original query type has been *query-ref*.

4.3 Nested & Interleaved Conversations

In this section, we extend the CP-net representation of previous sections to model nested and interleaved interaction protocols. We focus here on nested interaction protocols. Nevertheless, the discussion can also be addressed to interleaved interaction protocols in a similar fashion.

FIPA conversation standards [FIPA Specifications, 2005c] emphasize the importance of nested and interleaved protocols in modelling complex interactions. First, this allows re-use of interaction protocols in different nested interactions. Second, nesting increases the readability of interaction protocols.



(a) Nested protocol (b) Interleave protocol

Figure 4.12: AUML nested and interleaved protocols examples.

The AUML notation annotates nested and interleaved protocols as round corner rectangles [Odell *et al.*, 2001a, FIPA Specifications, 2005c]. Figure 4.12-a shows an example of a nested protocol, while Figure 4.12-b illustrates an interleaved protocol. Nested protocols have one or more compartments. The first compartment is the name compartment. The name compartment holds the (optional) name of the nested protocol. The nested protocol name is written in the upper left-hand corner of the rectangle, i.e. *commitment* in Figure 4.12-a. The second compartment, the guard compartment, holds the (optional) nested protocol guard. The guard compartment is written in the lower left-hand corner of the rectangle, e.g. [*commit*] in Figure 4.12-a. Nested protocols without guards are equivalent to nested protocols with the [*true*] guard.

Figure 4.13 describes the implementation of the nested interaction protocol presented in Figure 4.12-a by extending the CP-net representation to using hierarchies, relying on standard CP-net methods (see Appendix A). The hierarchical CP-net representation contains three elements: a *superpage*, a *subpage* and a *page hierarchy* graph. The CP-net superpage represents the main interaction protocol containing a nested interaction, while the CP-net subpage models the corresponding nested interaction protocol, i.e. the *Commitment Interaction Protocol*. The page hierarchy graph describes how the superpage is decomposed into subpages.

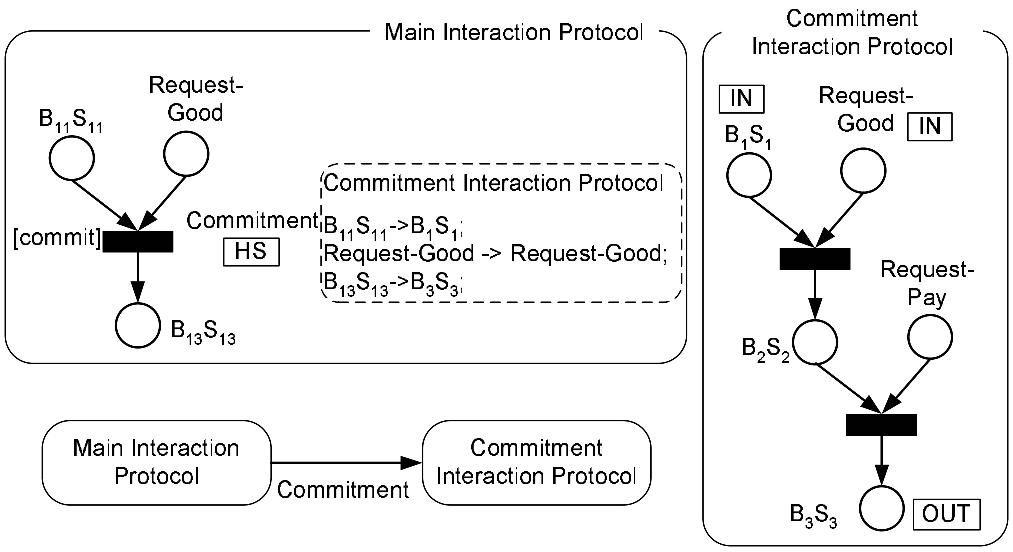

Main Interaction Protocol
Request-Good
$B_{11}S_{11}$
[commit]
Commitment
HS
Commitment Interaction Protocol
$B_{11}S_{11}$->B_1S_1;
Request-Good -> Request-Good;
$B_{13}S_{13}$->B_3S_3;
$B_{13}S_{13}$
Main Interaction Protocol
Commitment
Commitment Interaction Protocol
Commitment Interaction Protocol
IN
B_1S_1
Request-Good
IN
Request-Pay
B_2S_2
B_3S_3
OUT

Figure 4.13: Nested protocol implementation using hierarchical CP-nets.

Let us consider in detail the process of modelling the nested interaction protocol in Figure 4.12-a using a hierarchical CP-net, resulting in the net described in Figure 4.13. First, we identify the starting and ending points of the nested interaction protocol. The starting point of the nested interaction protocol is where

$Buyer_1$ sends a *Request-Good* communicative act to $Seller_1$. The ending point is where $Buyer_1$ receives a *Request-Pay* communicative act from $Seller_1$. We model these nested protocol end-points as CP-net *socket nodes* on the superpage, i.e. $Main\ Interaction\ Protocol$: $B_{11}S_{11}$ and *Request-Good* are input socket nodes and $B_{13}S_{13}$ is an output socket node.

The nested interaction protocol, the $Commitment\ Interaction\ Protocol$, is represented using a separate CP-net, following the principles outlined in Sections 4.1 and 4.2. This net is a subpage of the main interaction protocol superpage. The nested interaction protocol starting and ending points on the subpage correspond to the net *port nodes*. The B_1S_1 and *Request-Good* places are the subpage input port nodes, while the B_3S_3 place is an output port node. These nodes are tagged with the IN/OUT *port type tags* correspondingly.

Then, a *substitution transition*, which is denoted using HS (Hierarchy and Substitution), connects the corresponding socket places on the superpage. The substitution transition conceals the nested interaction protocol implementation from the net superpage, i.e. the $Main\ Interaction\ Protocol$. The nested protocol name and guard compartments are mapped directly to the substitution transition name and guard respectively. Consequently, in Figure 4.13 we define the substitution transition name as $Commitment$ and the substitution guard is determined to be [*commit*].

The superpage and subpage interface is provided using the hierarchy inscription. The hierarchy inscription is indicated using the dashed box next to the substitution transition. The first line in the hierarchy inscription determines the subpage identity, i.e. the $Commitment\ Interaction\ Protocol$ in our example. Moreover, it indicates that the substitution transition replaces the corresponding subpage detailed implementation on the superpage. The remaining hierarchy inscription lines introduce the superpage and subpage port assignment. The port assignment relates a socket node on the superpage with a port node on the subpage. The substitution transition input socket nodes are related to the IN-tagged port nodes. Analogously, the substitution transition output socket nodes correspond to the OUT-tagged port nodes. Therefore, the port assignment in Figure 4.13 assigns the net socket and port nodes in the following fashion: $B_{11}S_{11}$ to B_1S_1, *Request-Good* to *Request-Good* and $B_{13}S_{13}$ to B_3S_3.

Finally, the page hierarchy graph describes the decomposition hierarchy (nesting) of the different protocols (pages). The CP-net pages, the $Main\ Interaction\ Protocol$ and the $Commitment\ Interaction\ Protocol$, correspond to the page hierarchy graph nodes (Figure 4.13). The arc inscription indicates the substitution transition, i.e. *Commitment*.

4.4 Temporal Aspects of Conversations

Two temporal interaction aspects are specified by FIPA [FIPA Specifications, 2005c]. In this section, we show how *timed* CP-nets (see also Appendix A) can be applied for modelling agent interactions that involve temporal aspects, such as interaction duration, deadlines for message exchange, etc.

A first aspect, *duration*, is the interaction activity time period. Two periods can be distinguished: *transmission time* and *response time.* The transmission time indicates the time interval during which a communicative act, is sent by one agent and received by the designated receiver agent. The response time period denotes the time interval in which the corresponding receiver agent is performing some task as a response to the incoming communicative act.

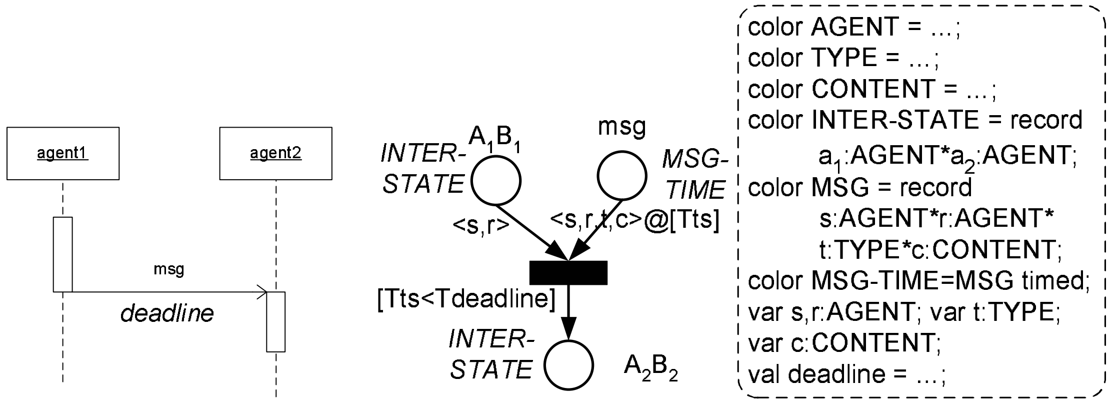
The second temporal aspect is *deadlines.* Deadlines denote the time limit by which a communicative act must be sent. Otherwise, the corresponding communicative act is considered to be invalid. These issues have not been addressed in previous investigations related to agent interactions modelling using Petri nets.[2]

We propose to utilize timed CP-nets techniques to represent these temporal aspects of agent interactions. In doing so, we assume a global clock.[3] We begin with deadlines. Figure 4.14-a introduces the AUML representation of message deadlines. The *deadline* keyword is a variation of the communicative act sequence expressions described in Section 4.2. It sets a time constraint on the start of the transmission of the associated communicative act. In Figure 4.14-a, $agent_1$ must send the *msg* communicative act to $agent_2$ before the defined *deadline*. Once the *deadline* expires, the *msg* communicative act is considered to be invalid.

Figure 4.14-b shows a timed CP-net implementation of the deadline sequence expression. The timed CP-net in Figure 4.14-b defines an additional *MSG-TIME* color set associated with the net message places. The *MSG-TIME* color set extends the MSG color set, described in Section 4.2, by adding a time stamp attribute to the

[2][Cost *et al.*, 1999, Cost *et al.*, 2000] mention deadlines without presenting any implementation details.

[3]Implementing it, we can use the private clock of an overhearing agent as the global clock for our Petri net representation. Thus, the time stamp of the message is the overhearer's time when the corresponding message was overheard.



(a) AUML representation (b) CP-net representation

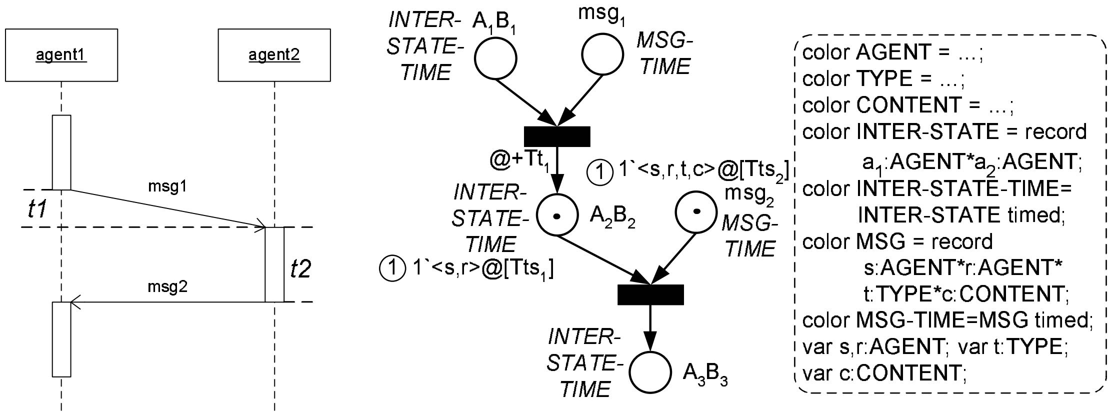
Figure 4.14: Deadline sequence expression.

message token. Thus, the communicative act token is a record $< s, r, t, c >$ @$[Tts]$. The @[..] expression denotes the corresponding token time stamp, whereas the token time value is indicated starting with a capital 'T'. Accordingly, the described message token has a *ts* time stamp. The communicative act time limit is defined using the val *deadline* parameter. Therefore, the deadline sequence expression semantics is simulated using the following transition guard: $[Tts < Tdeadline]$. This transition guard, comparing the *msg* time stamp against the *deadline* parameter, guarantees that an expired *msg* communicative act can not be received.

We now turn to representing interaction duration. The AUML representation is shown in Figure 4.15-a. The AUML time intensive message notation is used to denote the communicative act transmission time. As a rule communicative act arrows are illustrated horizontally. This indicates that the message transmission time can be neglected. However, in case the message transmission time is significant, the communicative act is drawn slanted downwards. The vertical distance, between the arrowhead and the arrow tail, denotes the message transmission time. Thus, the communicative act msg_1, sent from $agent_1$ to $agent_2$, has a t_1 transmission time.

The response time in Figure 4.15-a is indicated through the interaction thread length. The incoming msg_1 communicative act causes $agent_2$ to perform some task before sending a response msg_2 message. The corresponding interaction thread duration is denoted through the t_2 time period. Thus, this time period specifies the $agent_2$ response time to the incoming msg_1 communicative act.

The CP-net implementation to the interaction duration time periods is shown in Figure 4.15-b. The communicative act transmission time is illustrated using the



(a) AUML representation (b) CP-net representation

Figure 4.15: Interaction duration.

timed CP-nets @+ operator. The net transitions simulate the communicative act transmission between agents. Therefore, representing a transmission time of t_1, the CP-net transition adds a t_1 time period to the incoming message token time stamp. Accordingly, the transition $@+Tt_1$ output arc expression denotes a t_1 delay to the time stamp of the outgoing token. Thus, the corresponding transition takes t_1 time units and consequently so does the msg_1 communicative act transmission time.

In contrast to communicative act transmission time, the agent interaction response time is represented implicitly. Previously, we have defined a *MSG-TIME* color set that is amenable to indicate message token time stamps. Analogously, in Figure 4.15-b we introduce an additional *INTER-STATE-TIME* color set. This color set is associated with the net agent places and it presents the possibility to attach time stamps to agent tokens as well. Now, let us assume that A_2B_2 and msg_2 places contain a single token each. The circled '1' next to the corresponding place, together with the multi-set inscription, indicates the place current marking. Thus, the agent and the message place tokens have a ts_1 and a ts_2 time stamps respectively. The ts_1 time stamp denotes the time by which $agent_2$ has received the msg_1 communicative act sent by $agent_1$. The ts_2 time stamp indicates the time by which $agent_2$ is ready to send msg_2 response message to $agent_1$. Thus, the $agent_2$ response time t_2 (Figure 4.15-a) is $ts_2 - ts_1$.

4.5 A Complex Conversation Protocol

In this section, we present an example of a complex *3-agent* AUML conversation protocol modelled using our CP-net representation. This example incorporates many advanced features of our CP-net representation technique and would have been beyond the scope of many previous investigations.

The conversation protocol addressed here is the *FIPA Brokering Interaction Protocol* [FIPA Specifications, 2005a]. This interaction protocol incorporates many advanced conversation features of our representation such as nesting, communicative act sequence expression, message guards and etc. Its AUML representation is shown in Figure 4.16.

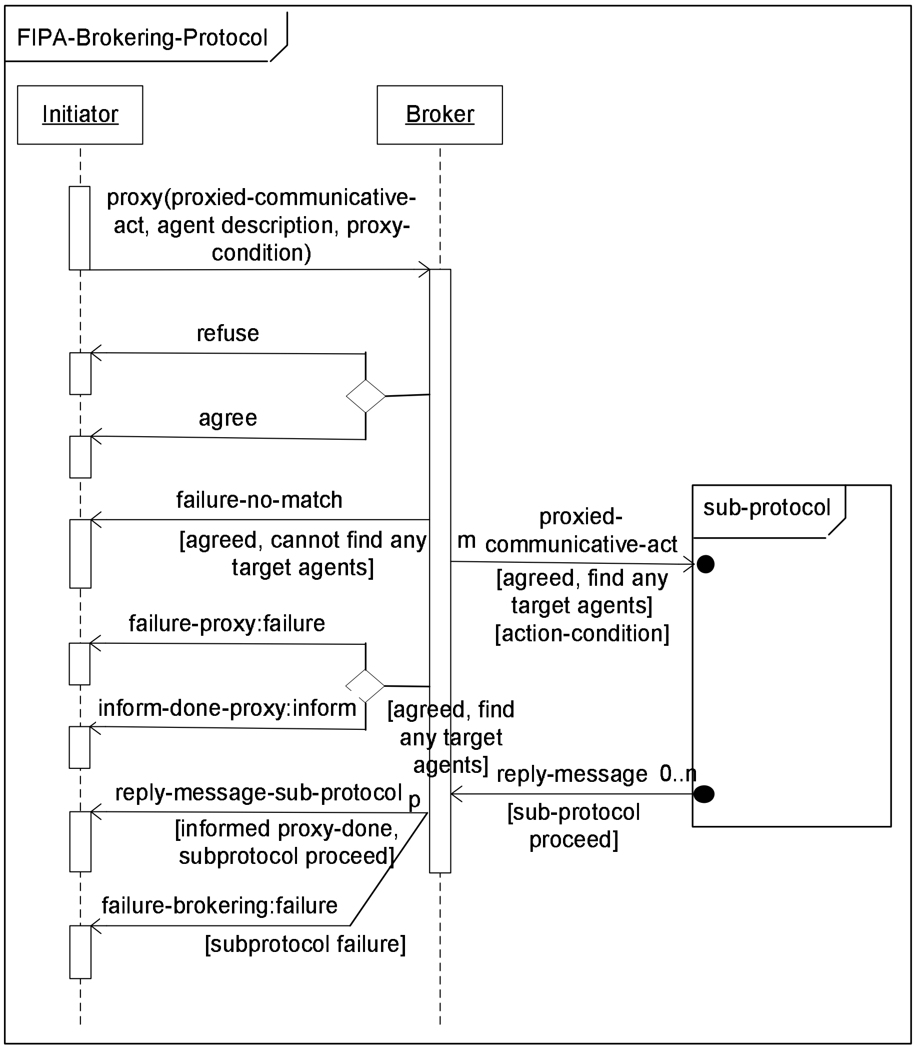

FIPA-Brokering-Protocol
Initiator
Broker
proxy(proxied-communicative-act, agent description, proxy-condition)
refuse
agree
failure-no-match
[agreed, cannot find any target agents]
failure-proxy:failure
inform-done-proxy:inform
[agreed, find any target agents]
reply-message-sub-protocol
p
[informed proxy-done, subprotocol proceed]
failure-brokering:failure
[subprotocol failure]
m
proxied-communicative-act
[agreed, find any target agents]
[action-condition]
reply-message 0..n
[sub-protocol proceed]
sub-protocol

Figure 4.16: FIPA Brokering Interaction Protocol - AUML representation.

The *Initiator* agent begins the interaction by sending a *proxy* message to the *Broker* agent. The *proxy* communicative act contains the requested communicative act, i.e. *proxied-communicative-act*, as part of its argument list. The *Broker* agent processes the request and responds with either an *agree* or a *refuse* message. Communication of a *refuse* message terminates the interaction. If the *Broker* agent has agreed to function as a proxy, it then locates the agents matching the *Initiator* request. If no such agent can be found, the *Broker* agent communicates a *failure-no-match* message and the interaction terminates. Otherwise, the *Broker* agent begins m interactions with the matching agents. For each such agent, the *Broker* informs the *Initiator*, sending either an *inform-done-proxy* or a *failure-proxy* communicative act. The *failure-proxy* communicative act terminates the *sub-protocol* interaction with the matching agent in question. The *inform-done-proxy* message continues the interaction. As the *sub-protocol* progresses, the *Broker* forwards the received responses to the *Initiator* agent using the *reply-message-sub-protocol* communicative acts. However, there can be other failures that are not explicitly returned from the *sub-protocol* interaction (e.g., if the agent executing the *sub-protocol* has failed). In case the *Broker* agent detects such a failure, it communicates a *failure-brokering* message, which terminates the *sub-protocol* interaction.

A CP-net representation of the *FIPA Brokering Interaction Protocol* is shown in Figure 4.17. The *Brokering Interaction Protocol* starts from I_1B_1 place. The I_1B_1 place represents a joint interaction state where *Initiator* is ready to send a *proxy* communicative act (I_1) and *Broker* is waiting to receive it (B_1). The *proxy* communicative act causes the interacting agents to transition to I_2B_2. This place denotes an interaction state in which *Initiator* has already sent a *proxy* message to *Broker* (I_2) and *Broker* has received it (B_2). The *Broker* agent can send, as a response, either a *refuse* or an *agree* communicative act. This CP-net component is implemented using the XOR-decision building block presented in Section 4.1. The *refuse* message causes the agents to transition to I_3B_3 place and thus terminate the interaction. This place corresponds to *Broker* sending a *refuse* message and terminating (B_3), while *Initiator* receiving the message and terminating (I_3). On the other hand, the *agree* communicative act causes the agents to transition to I_4B_4 place, which represents a joint interaction state in which the *Broker* has sent an *agree* message to *Initiator* (and is now trying to locate the receivers of the *proxied* message), while the *Initiator* received the *agree* message.

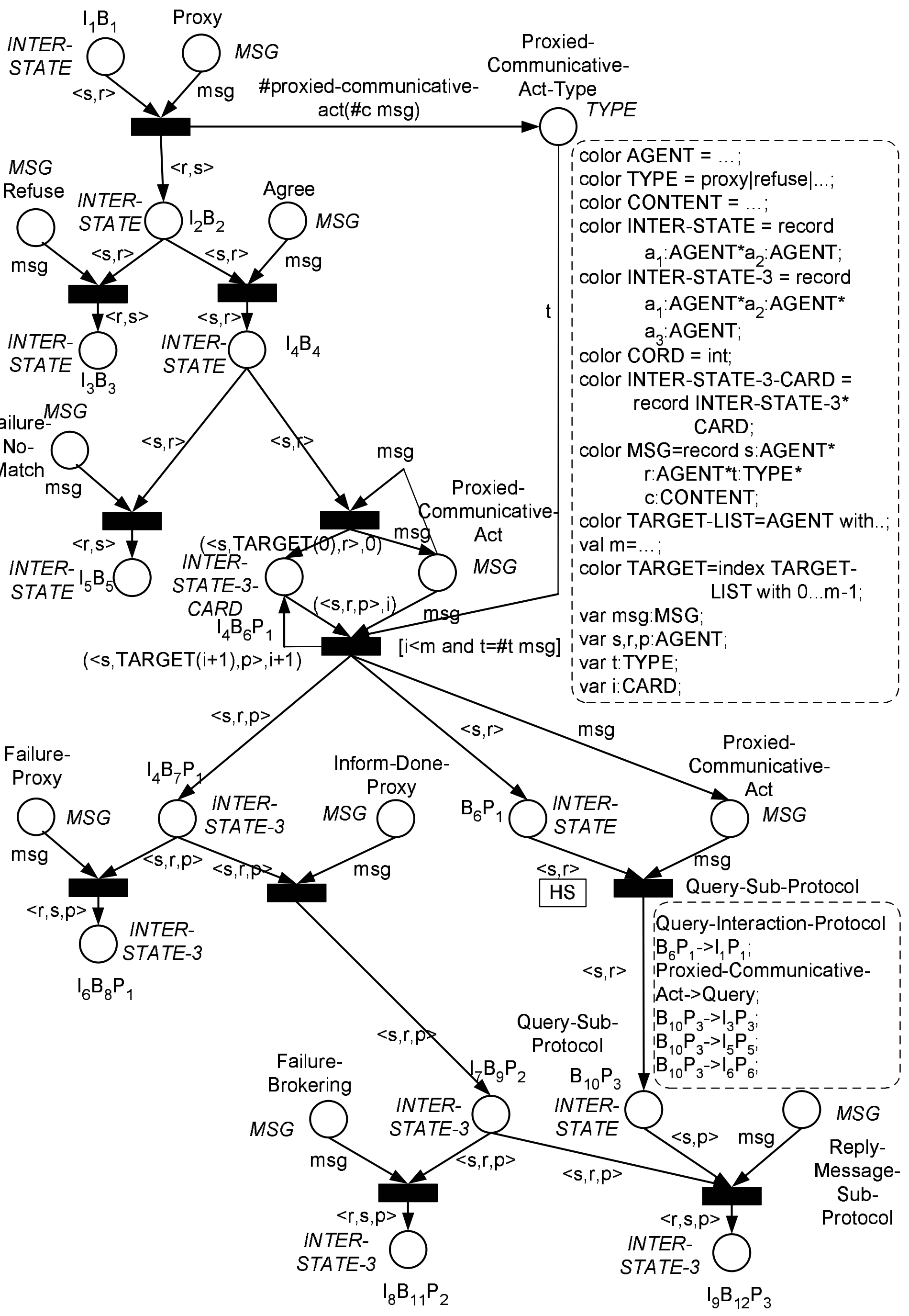


Figure 4.17: FIPA Brokering Interaction Protocol - CP-net representation.

The *Broker* agent's search for suitable receivers may result in two alternatives. First, in case no matching agents are found, the interaction terminates in the I_5B_5 agent place. This joint interaction place corresponds to an interaction state where *Broker* has sent the *failure-no-match* communicative act (B_5), and *Initiator* has received the message and terminated (I_5). The second alternative is that suitable agents have been found. Then, *Broker* starts sending *proxied-communicative-act* messages to these agents on the established list of designated receivers, i.e. *TARGET-LIST*. The first such *proxied-communicative-act* message causes the interacting agents to transition to $I_4B_6P_1$ place. The $I_4B_6P_1$ place denotes a joint interaction state of three agents: *Initiator*, *Broker* and *Participant* (the receiver). The *Initiator* individual state remains unchanged (I_4) since the *proxied-communicative-act* message starts an interaction between *Broker* and *Participant*. The *Broker* individual state (B_6) denotes that designated agents have been found and the *proxied-communicative-act* messages are ready to be sent, while *Participant* is waiting to receive the interaction initiating communicative act (P_1). The *proxied-communicative-act* message place is also connected as an output place of the transition. This message place is used as part of a CP-net XOR-decision structure, which enables the *Broker* agent to send either a *failure-no-match* or a *proxied-communicative-act*, respectively. Thus, the token denoting the *proxied-communicative-act* message, must not be consumed by the transition.

Thus, multiple *proxied-communicative-act* messages are sent to all *Participants*. This is implemented similarly to the broadcast sequence expression implementation (Section 4.2). Furthermore, the *proxied-communicative-act* type is verified against the type of the requested *proxied* communicative act, which is obtained from the original proxy message content. We use the *Proxied-Communicative-Act-Type* message type place to implement this CP-net component similarly to Figure 4.11. Each *proxied-communicative-act* message causes the interacting agents to transition to both the $I_4B_7P_1$ and the B_6P_1 places.

The B_6P_1 place corresponds to interaction between the *Broker* and the *Participant* agents. It represents a joint interaction state in which *Broker* is ready to send a *proxied-communicative-act* message to *Participant* (B_6), and *Participant* is waiting for the message (P_1). In fact, the B_6P_1 place initiates the nested interaction protocol that results in $B_{10}P_3$ place. The $B_{10}P_3$ place represents a joint interaction state where *Participant* has sent the *reply-message* communicative act and terminated (P_3), and *Broker* has received the message (B_{10}). In our example, we have chosen the *FIPA Query Interaction Protocol* [FIPA Specifications, 2005d] (Figures 4.10–4.11) as the interaction *sub-protocol*. The CP-net component, implementing the nested interaction *sub-protocol*, is modelled using the principles described in Section 4.3. Consequently, the interaction *sub-protocol* is concealed using the *Query-Sub-Protocol* substitution transition. The B_6P_1, *proxied-communicative-act* and $B_{10}P_3$ places determine substitution transition socket nodes. These socket nodes are assigned to the CP-net port nodes in Figure 4.11 as follows. The B_6P_1 and *proxied-communicative-act* places are assigned to the I_1P_1 and *query* input port nodes, while the $B_{10}P_3$ place is assigned to the I_3P_3, I_5P_5 and I_6P_6 output port nodes.

We now turn to the $I_4B_7P_1$ place. In contrast to the B_6P_1 place, this place corresponds to the main interaction protocol. The $I_4B_7P_1$ place represents a joint interaction state in which *Initiator* is waiting for *Broker* to respond (I_4), *Broker* is ready to send an appropriate response communicative act (B_7), and to the best of the *Initiator*'s knowledge the interaction with *Participant* has not yet begun (P_1). The *Broker* agent can send one of two messages, either a *failure-proxy* or an *inform-done-proxy*, depending on whether it has succeeded to send the *proxied-communicative-act* message to *Participant*. The *failure-proxy* message causes the agents to terminate the interaction with corresponding *Participant* agent and to transition to $I_6B_8P_1$ place. This place denotes a joint interaction state in which *Initiator* has received a *failure-proxy* communicative act and terminated (I_6), *Broker* has sent the *failure-proxy* message and terminated as well (B_8) and the interaction with the *Participant* agent has never started (P_1). On the other hand, the *inform-done-proxy* causes the agents to transition to $I_7B_9P_2$ place. The $I_7B_9P_2$ place represents an interaction state where *Broker* has sent the *inform-done-proxy* message (B_9), *Initiator* has received it (I_7), and *Participant* has begun the interaction with the *Broker* agent (P_2). Again, this is represented using the XOR-decision building block.

Finally, the *Broker* agent can either send a *reply-message-sub-protocol* or a *failure-brokering* communicative act. The *failure-brokering* message causes the interacting agents to transition to $I_8B_{11}P_2$ place. This place indicates that *Broker* has sent a *failure-brokering* message and terminated (B_{11}), *Initiator* has received the message and terminated (I_8), and *Participant* has terminated during the interaction with the *Broker* agent (P_2). The *reply-message-sub-protocol* communicative act causes the agents to transition to $I_9B_{12}P_3$ place. The $I_9B_{12}P_3$ place indicates that *Broker* has sent a *reply-message-sub-protocol* message and terminated (B_{12}), *Initiator* has received the message and terminated (I_9), and *Participant* has successfully completed the nested *sub-protocol* with the *Broker* agent and terminated as well (P_3). Thus, the $B_{10}P_3$ place, denoting a successful completion of the nested *sub-protocol*, is also the corresponding transition input place.

Chapter 5

Transforming AUML Diagrams to CP-net Representation

Our final contribution in this part of the thesis is a skeleton procedure for transforming a human-readable AUML conversation protocol diagram of two interacting agents to its machine-readable CP-net representation (Section 5.1). The procedure is semi-automated–it relies on the human to fill in some details–but also has automated aspects. We apply this procedure on a complex multi-agent conversation protocol that involves many of the interaction building blocks already discussed (Section 5.2).

5.1 Transformation Algorithm

The procedure is shown in Algorithm 1. The algorithm input is an AUML protocol diagram and the algorithm creates, as an output, a corresponding CP-net representation. The CP-net is constructed in iterations using a queue. The algorithm essentially creates the conversation net by exploring the interaction protocol breadth-first while avoiding cycles.

Lines 1-2 create and initiate the algorithm queue, and the output CP-net, respectively. The queue, denoted by S, holds the initiating agent places of the current iteration. These places correspond to interaction states that initiate further conversation between the interacting agents. In lines 4-5, an initial agent place A_1B_1 is created and inserted into the queue. The A_1B_1 place represents a joint initial interaction state for the two agents. Lines 7-23 contain the main loop.

We enter the main loop in line 8 and set the *curr* variable to the first initiating agent place in S queue. Lines 10-13 create the CP-net components corresponding

Algorithm 1 Create Conversation Net(**input**:$AUML$,**output**:CPN)

```
1:  $S \leftarrow$ **new** queue
2:  $CPN \leftarrow$ **new** $\mathrm{CP-net}$
3:
4:  $A_1B_1 \leftarrow$ **new** agent place with color information
5:  $S.enqueue(A_1B_1)$
6:
7:  **while** $S$ **not** *empty* **do**
8:      $curr \leftarrow S.dequeue()$
9:
10:     $MP \leftarrow CreateMessagePlaces(AUML, curr)$
11:     $RP \leftarrow CreateResultingAgentPlaces(AUML, curr, MP)$
12:     $(TR, AR) \leftarrow CreateTransitionsAndArcs(AUML, curr, MP, RP)$
13:     $FixColor(AUML, CPN, MP, RP, TR, AR)$
14:
15:     **for each** *place* $p$ **in** $RP$ **do**
16:         **if** $p$ was not created in current iteration **then**
17:             **continue**
18:         **if** $p$ **is not** terminating place **then**
19:             $S.enqueue(p)$
20:
21:     $CPN.places = CPN.places \bigcup MP \bigcup RP$
22:     $CP.transitions = CPN.transitions \bigcup TR$
23:     $CPN.arcs = CPN.arcs \bigcup AR$
24:
25: **return** $CPN$
```

to the current iteration as follows. First, in line 10, message places, associated with *curr* agent place, are created using the $CreateMessagePlaces$ procedure. This procedure extracts the communicative acts that are associated with a given interaction state, from the AUML diagram. These places correspond to communicative acts, which take agents from the joint interaction state *curr* to its successor(s). Then in line 11, the $CreateResultingAgentPlaces$ procedure creates agent places that correspond to interaction state changes as a result of the communicative acts associated with *curr* agent place (again based on the AUML diagram). Then, in $CreateTransitionsAndArcs$ procedure (line 12), these places are connected using the principles described in Sections 4.1–4.4 of Chapter 4. Thus, the CP-net structure (net places, transitions and arcs) is created. Finally, in line 13, the *FixColor* procedure adds token color elements to the CP-net structure, to support deadlines, cardinality, and other communicative act attributes.

Lines 15-19 determine which resulting agent places are inserted into the S queue for further iteration. Only non-terminating agent places, i.e. places that do not correspond to interaction states that terminate the interaction, are inserted into the queue in lines 18-19. However, there is one exception (lines 16-17): a resulting agent place, which has already been handled by the algorithm, is not inserted back into the S queue since inserting it can cause an infinite loop. Thereafter, completing the current iteration, the output CP-net, denoted by CPN variable, is updated according to the current iteration CP-net components in lines 21-23. This main loop iterates as long as the S queue is not empty. The resulting CP-net is returned– line 25.

5.2 Transformation Example

To demonstrate this algorithm, we will now use it on the *FIPA Contract Net Interaction Protocol* [FIPA Specifications, 2005b] (Figure 5.1). This protocol allows interacting agents to negotiate. The $Initiator$ agent issues m calls for proposals using a cfp communicative act. Each of the m *Participants* may refuse or counter-propose by a given *deadline* sending either a $refuse$ or a *propose* message respectively. A $refuse$ message terminates the interaction. In contrast, a *propose* message continues the corresponding interaction.

Once the *deadline* expires, the $Initiator$ does not accept any further $Participant$ response messages. It evaluates the received $Participant$ proposals and selects one, several, or no agents to perform the requested task. Accepted proposal result in the sending of *accept-proposal* messages, while the remaining proposals are rejected using *reject-proposal* message. *Reject-proposal* terminates the interaction with the corresponding $Participant$. On the other hand, the *accept-proposal* message commits a $Participant$ to perform the requested task. On successful completion, $Participant$ informs $Initiator$ sending either an *inform-done* or an *inform-result* communicative act. However, in case a $Participant$ has failed to accomplish the task, it communicates a $failure$ message.

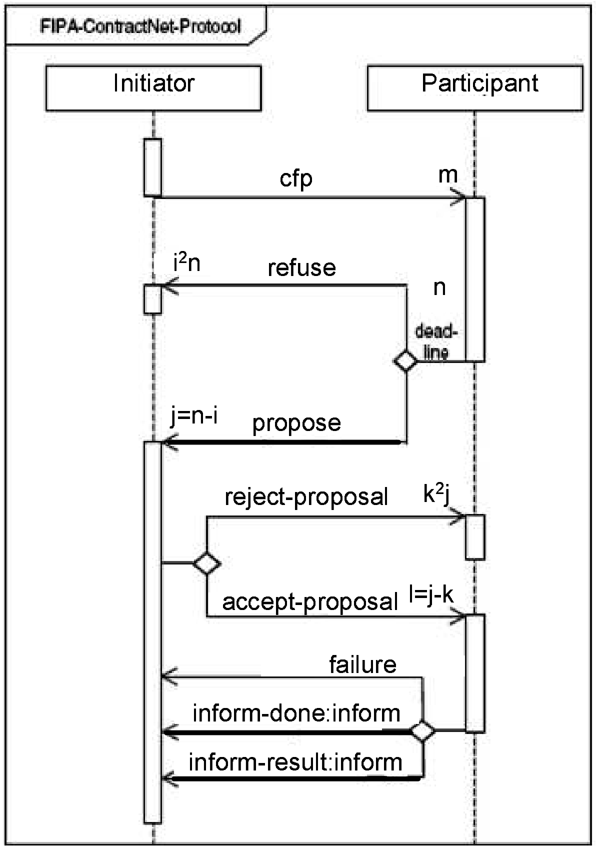


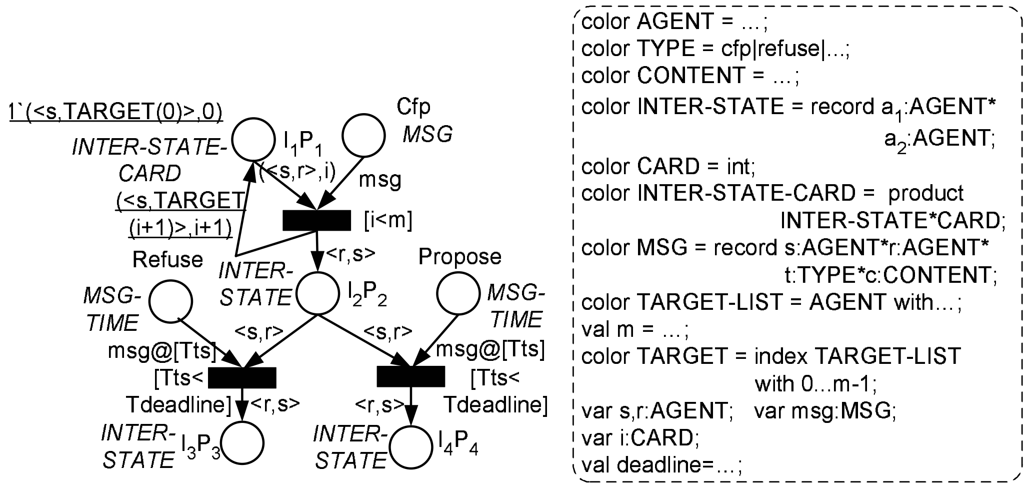
Figure 5.1: FIPA Contract Net Interaction Protocol using AUML.

We now use the algorithm introduced above to create a CP-net, which represents the *FIPA Contract Net Interaction Protocol*. The corresponding CP-net model is constructed in four iterations of the algorithm. Figure 5.2 shows the CP-net representation after the second iteration of the algorithm, while Figure 5.3 shows the CP-net representation after the fourth and final iteration.

The *Contract Net Interaction Protocol* starts from I_1P_1 place, which represents a joint interaction state where *Initiator* is ready to send a cfp communicative act (I_1) and *Participant* is waiting for the corresponding cfp message (P_1). The I_1P_1 place is created and inserted into the queue before the iterations through the main loop begin.

First iteration. The *curr* variable is set to the I_1P_1 place. The algorithm creates net places, which are associated with the I_1P_1 place, i.e. a Cfp message place, and an I_2P_2 resulting agent place. The I_2P_2 place denotes an interaction state in which *Initiator* has already sent a cfp communicative act to *Participant* and is now waiting for its response (I_2) and *Participant* has received the cfp message and is now deciding on an appropriate response (P_2). These are created using the *CreateMessagePlaces* and the *CreateResultingAgentPlaces* procedures, respectively.

Then, the *CreateTransitionsAndArcs* procedure in line 12, connects the three places using a simple asynchronous message building block as shown in Figure 4.1-b (Section 4.1). In line 13, as the color sets of the places are determined, the algorithm also handles the cardinality of the *cfp* communicative act, by putting an appropriate sequence expression on the transition, using the principles presented in Figure 4.9-b (Section 4.2). Accordingly, the color set, associated with I_1P_1 place, is changed to the *INTER-STATE-CARD* color set. Since the I_2P_2 place is not a terminating place, it is inserted into the S queue.



```
color AGENT = ...;
color TYPE = cfp|refuse|...;
color CONTENT = ...;
color INTER-STATE = record a1:AGENT*
                           a2:AGENT;
color CARD = int;
color INTER-STATE-CARD = product
                   INTER-STATE*CARD;
color MSG = record s:AGENT*r:AGENT*
                t:TYPE*c:CONTENT;
color TARGET-LIST = AGENT with...;
val m = ...;
color TARGET = index TARGET-LIST
              with 0...m-1;
var s,r:AGENT;   var msg:MSG;
var i:CARD;
val deadline=...;
```

Figure 5.2: FIPA Contract Net Interaction Protocol using CP-net after the 2nd iteration.

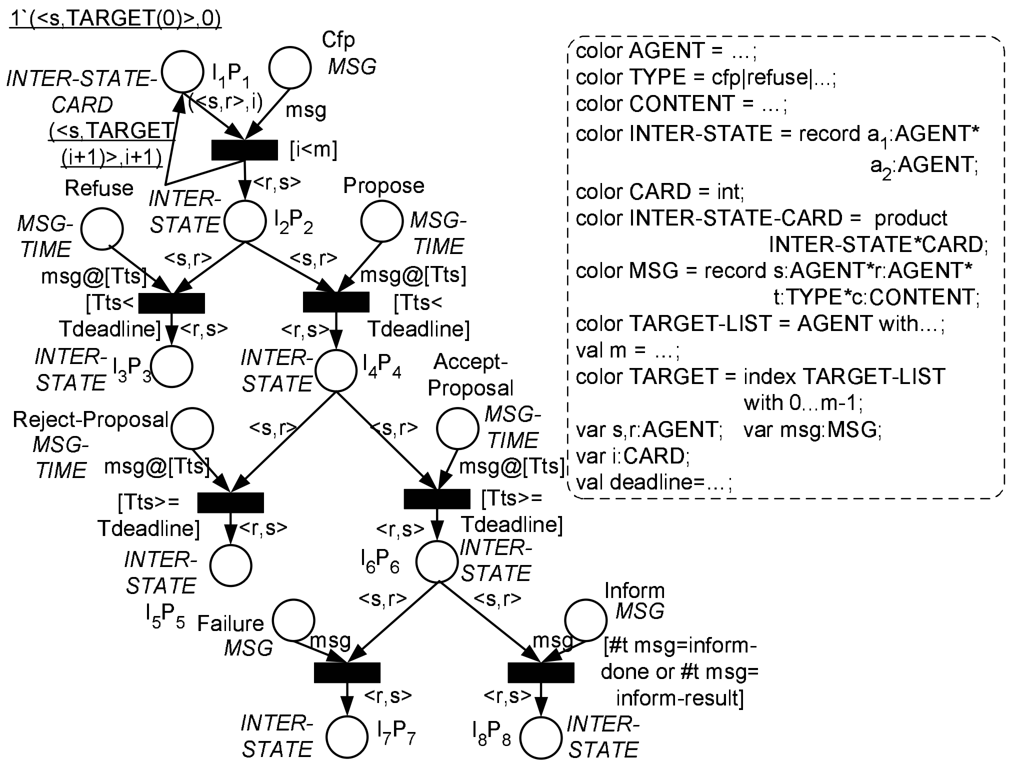
Second iteration. *curr* is set to the I_2P_2 place. The *Participant* agent can send, as a response, either a *refuse* or a *propose* communicative act. *Refuse* and *Propose* message places are created by *CreateMessagePlaces* (line 10), and resulting places I_3P_3 and I_4P_4, corresponding to the results of the *refuse* and *propose* communicative acts, respectively, are created by *CreateResultingAgentPlaces* (line 11). The I_3P_3 place represents a joint interaction state where *Participant* has sent the *refuse* message and terminated (P_3), while *Initiator* has received it, and terminated (I_3). The I_4P_4 place represents the joint state in which *Participant* has sent the *propose* message (P_4), while *Initiator* has received the message and is considering its response (I_4).

In line 12, the I_2P_2, $Refuse$, I_3P_3, *Propose* and I_4P_4 places are connected using the XOR-decision building block presented in Figure 4.3-b (Section 4.1). Then, the $FixColor$ procedure (line 13), adds the appropriate token color attributes, to allow a deadline sequence expression (on both the $refuse$ and the *propose* messages) to be implemented as shown in Figure 4.14-b (Section 4.4). The I_3P_3 place denotes a terminating state, whereas the I_4P_4 place continues the interaction. Thus, in lines 18-19, only the I_4P_4 place is inserted into the queue, for the next iteration of the algorithm. The state of the net at the end of the second iteration of the algorithm is presented in Figure 5.2.

Third iteration. *curr* is set to I_4P_4. Here, the $Initiator$ response to a $Participant$ proposal can either be an *accept-proposal* or a *reject-proposal*. *CreateMessagePlaces* procedure in line 10 thus creates the corresponding *Accept-Proposal* and *Reject-Proposal* message places. The *accept-proposal* and *reject-proposal* messages cause the interacting agents to transition to I_5P_5 and I_6P_6 places, respectively. These agent places are created using the $CreateResultingAgentPlaces$ procedure (line 11). The I_5P_5 place denotes an interaction state in which $Initiator$ has sent a *reject-proposal* message and terminated the interaction (I_5), while the $Participant$ has received the message and terminated as well (P_5). In contrast, the I_6P_6 place represents an interaction state where $Initiator$ has sent an *accept-proposal* message and is waiting for a response (I_6), while $Participant$ has received the *accept-proposal* communicative act and is now performing the requested task before sending a response (P_6). The $Initiator$ agent sends exclusively either an *accept-proposal* or a *reject-proposal* message. Thus, the I_4P_4, *Reject-Proposal*, I_5P_5, *Accept-Proposal* and I_6P_6 places are connected using a XOR-decision block (in the $CreateTransitionsAndArcs$ procedure, line 12).

The $FixColor$ procedure in line 13 operates now as follows: According to the interaction protocol semantics, the *Initiator* agent evaluates all the received *Participant* proposals once the *deadline* passes. Only thereafter, the appropriate *reject-proposal* and *accept-proposal* communicative acts are sent. Thus, $FixColor$ assigns a *MSG-TIME* color set to the *Reject-Proposal* and the *Accept-Proposal* message places, and creates a $[Tts >= Tdeadline]$ transition guard on the associated transitions. This transition guard guarantees that $Initiator$ cannot send any response until the *deadline* expires, and all valid $Participant$ responses have been received.

The resulting I_5P_5 agent place denotes a terminating interaction state, whereas the I_6P_6 agent place continues the interaction. Thus, only I_6P_6 agent place is inserted into the S queue.



```
color AGENT = ...;
color TYPE = cfp|refuse|...;
color CONTENT = ...;
color INTER-STATE = record a1:AGENT*
                                      a2:AGENT;
color CARD = int;
color INTER-STATE-CARD =  product
                          INTER-STATE*CARD;
color MSG = record s:AGENT*r:AGENT*
                          t:TYPE*c:CONTENT;
color TARGET-LIST = AGENT with...;
val m = ...;
color TARGET = index TARGET-LIST
                      with 0...m-1;
var s,r:AGENT;   var msg:MSG;
var i:CARD;
val deadline=...;
```

Figure 5.3: FIPA Contract Net Interaction Protocol using CP-net after the 4th (and final) iteration.

Fourth iteration. *curr* is set to I_6P_6. This place is associated with three communicative acts: *inform-done*, *inform-result* and $failure$. The *inform-done* and the *inform-result* messages are instances of the $inform$ communicative act class. Thus, $CreateMessagePlaces$ (line 10) creates only two message places, $Inform$ and $Failure$. In line 11, $CreateResultingAgentPlaces$ creates the I_7P_7 and I_8P_8 agent places. The $failure$ communicative act causes interacting agents to transition to I_7P_7 agent place, while both $inform$ messages cause the agents to transition to I_8P_8 agent place. The I_7P_7 place represents a joint interaction state where $Participant$ has sent the $failure$ message and terminated (P_7), while $Initiator$ has received a $failure$ communicative act and terminated (I_7). On the other hand, the I_8P_8 place denotes an interaction state in which $Participant$ has sent the $inform$ message (either *inform-done* or *inform-result*) and terminated

(P_8), while $Initiator$ has received an $inform$ communicative act and terminated (I_8). The $inform$ and $failure$ communicative acts are sent exclusively. Thus $CreateTransitionsAndArcs$ (line 12) connects the I_6P_6, $Failure$, I_7P_7, $Inform$ and I_8P_8 places using a XOR-decision building block. Then, $FixColor$ assigns a [*#t msg = inform-done* or *#t msg = inform-result*] transition guard on the transition associated with $Inform$ message place. Since both the I_7P_7 and the I_8P_8 agent places represent terminating interaction states, they are not inserted into the queue, which remains empty at the end of the current iteration. This signifies the end of the conversion. The complete conversation CP-net resulting after this iteration of the algorithm is shown in Figure 5.3.

The procedure we outline can guide the conversion of many *2-agent* conversation protocols in AUML to their CP-net equivalents. However, it is not sufficiently developed to address the general *n-agent* case. For instance, Section 4.5 presented a complex example of a *3-agent* conversation protocol, which was successfully converted manually, but could not be converted using the present form of the algorithm. We believe that addressing automated conversion of the general *n-agent* case conversation protocols can be a potential ground for future research.

Part II

Conversation Recognition

Previous investigations on overhearing demonstrated a range of overhearing techniques as presented in Chapter 2. However, despite the inspiration and concrete techniques provided by previous work, general challenges of overhearing were only addressed in the context of specific applications. As a result, a formal model of overhearing was yet to be presented.

In this part of the thesis, we address this challenge introducing a formal approach to overhearing (Chapter 6). We present a comprehensive theoretical model that is constructed from three components. The first models the representation of conversation protocols, i.e. inter-agent communication templates used to coordinate a specific system task performance (e.g., FIPA interaction protocols [FIPA site, 2005]). The second component models a complete conversation system, a set of instantiated conversations that take place in a multi-agent system. Finally, the third component of our model represents the view of an overhearing agent on the corresponding conversation system.

In addition, some key assumptions made by previous works are difficult to extract. For instance, previous investigations on overhearing all make the assumption that the overhearing agent can match intercepted messages to a conversation protocol. Most make the assumption that all messages in a conversation are overheard (i.e. no losses). Yet both assumptions are challenged in real-world settings.

Our work seeks to address these assumptions by presenting conversation recognition algorithms (Chapter 7). We use the proposed theoretical model to formulate the problem of conversation recognition. Conversation recognition is a key step in overhearing, that deals with recognizing the conversation that took place, given a set of overheard messages. This is a preliminary step to obtaining information from overheard conversations.

We provide a skeleton algorithm for this task and instantiate it for handling lossless and lossy overhearing. We explore the complexity of these algorithms to address their appropriateness for large-scale settings, and show that handling general lossy overhearing–overhearing where messages can randomly be lost–is computationally expensive. Surprisingly, however, a specific case of lossy overhearing, called systematic message loss, e.g., always losing one side of the conversation, is significantly more efficient in terms of complexity. Fortunately, systematic message loss is likely to be more frequent in practice.

Chapter 6

A Formal Model of Overhearing

Addressing the general overhearing task, we propose a formal model that is constructed of three components: (i) conversation protocols; (ii) a system of conversations using conversation protocols; and (iii) a view on conversations by an overhearing agent.

To demonstrate the proposed model, we consider the following overheard conversation between two agents bidding on a contract. The first agent sends a call for proposal (*cfp*), on which the second agent replies with a proposal–a *propose* message. Then, the first agent accepts this proposal by sending an *accept-proposal*. Finally, the second agent performs the agreed task and communicates an *inform* message–informing the first agent on the established results.

This conversation implements a portion of the *FIPA Contract Net Interaction Protocol* [FIPA Specifications, 2005b]. Generally[1], the same protocol can be overheard differently. After the first agent issues a *cfp*, a second agent can *refuse* it or *propose* to it. Then, its proposal is either accepted or rejected by the first agent-communicating an *accept-proposal* or a *reject-proposal* message. Finally, the second agent notifies the first agent on the results of the performed task sending an *inform* or a *failure* message.

In the following sub-sections, we discuss the various components of our model demonstrating them using the presented protocol and conversation.

[1]We describe this pattern as it may appear to the overhearing agent.

6.1 Conversation Protocols

When involved in a conversation, agents normally communicate according to a protocol, which can be captured by well-defined patterns. These patterns, i.e. *conversation protocols*, define a template that conversations must follow to achieve a communications goal. Hence, conversation protocols specify an abstract representation of the corresponding conversations.

Conversation protocols are widely used in open multi-agent settings. For instance, FIPA protocols [FIPA site, 2005] are an example to the continuous effort to standardize the use of conversation protocols in multi-agent community. Though frequently used in agent-oriented settings, conversation protocols can be found in human-oriented environments as well. For example, [McElhearn, 1996] has showed that conversation protocols can be found in e-mail mailing list traffic.

In our model, a conversation protocol is a tuple denoted by $(R, \Sigma, S, s_0, F, \delta)$. Below, we provide a detailed discussion of the components of this tuple.

Conversation Roles (R): A conversation role defines a separate functionality in a conversation. Conversation protocols define valid sequences of messages between various conversation roles. Each role determines agent behavior in a specific conversation. In our model, R denotes the set of conversation roles in a conversation protocol. In the contract-net protocol shown above, two roles can be distinguished: the first agent is the *initiator*, whereas the second agent is the *participant* [FIPA Specifications, 2005b]. Thus, the set R consists of these two conversation roles.

Communicative Act Types (Σ): There are often multiple communicative act types (e.g. in FIPA [FIPA Communicative Acts, 2005]). Σ denotes the set of all communicative act types used by the given conversation protocol. In our example, this set contains: cfp, $refuse$, *propose*, etc.

Conversation States (S): A conversation state of an agent marks its state within the protocol (in contrast with its internal state). Here, we must distinguish between *individual* and *joint* states (see Part I, Chapter 3 for more details).

We explain these terms using the contract-net protocol. We denote the two conversation roles as A and B. For the moment, let us consider A individually. The A role starts in an initial conversation state, denoted as A_1, where initiator is ready to send a cfp message type. Sending this message, the agent transitions to its second conversation state (A_2) in which it has already sent a cfp type message and is now waiting to receive either a *propose* or a $refuse$ message type. Receiving it, the agent transitions to one of the A_3 or A_4 conversation states, and so on. Similarly individual conversation states A_1-A_8 and B_1-B_8 can be defined over the A and B roles respectively. Here, we do not present a detailed discussion on the protocol implementation, but we refer the readers to Part I, Chapter 5, Section 5.2 for additional information.

The same protocol may also be defined using a collection of joint interaction states (Part I, Chapter 3), $S = S_A \times S_B$, where each member of S corresponds to a specific combination of individual states. However, not all joint states are legal. For example, A_1B_1 is a legal joint conversation state in the given conversation protocol, whereas A_2B_1 joint conversation state is considered to be illegal (since it denotes a state where A has sent a message but B did not receive it). In our model, S is the set of all legal joint conversation states over a conversation protocol. In our example, S contains the following joint conversation states: A_1B_1, A_2B_2, etc.

Initiating Conversation State (s_0): s_0 is an initiating joint conversation state, which corresponds to the combination of the initiating individual conversation states over the various conversation roles.

Terminating Conversation States (F): F defines the set of joint conversation states that terminate the conversation. Thus, $F \subseteq S$. In our example, F includes A_3B_3 joint conversation state in which the *initiator* received a $refuse$ message and terminated, whereas the *participant* has sent it and terminated as well.

Transition Function (δ): δ determines the progress of a conversation by defining which message types are expected at different points of the conversation according to its current conversation state.

In order to define δ, we must first define following parameters. An abstract message am is a $< r_x, r_y, \sigma >$, which is a member of the relation AM, where $AM = \{< r_x, r_y, \sigma > | r_x, r_y \in R,\ \sigma \in \Sigma \text{ and } r_x \neq r_y\}$. Thus, AM denotes a set of abstract messages that may potentially correspond to the appropriate conversation protocol.

Now, we define δ as $S \times AM \rightarrow S$. Thus, the δ function defines whether a transition, between two legal joint conversation states, is possible. In addition, δ determines the abstract message, which causes this transition to occur.

In the example above, let us consider the $\delta(A_1B_1, < A, B, cfp >) = A_2B_2$ instance of δ. This instance has the following interpretation: given agents in A_1B_1

joint conversation state, the $< A, B, cfp >$ abstract message (of a cfp message type sent from the *initiator* to the *participant*) causes the agents to transition to the A_2B_2 joint conversation state.

Based on the definition of conversation protocols presented above, we can now define the set of possible abstract conversation sequences over a conversation protocol. Given a conversation protocol p, we denote this set as $AS(p)$. To define this parameter, we define a transition function on a sequence of abstract messages. This function, defined as $\delta^* : S \times AM^* \rightarrow S$ (where AM^* denotes the set of all possible sequences over AM), can be formulated recursively as follows:

$$\begin{aligned}
&\delta^*(s, \varepsilon) = \varepsilon \\
&\delta^*(s, \omega\nu) = \delta(\delta^*(s, \omega), \nu) \\
&\text{where } s \in S,\ \varepsilon(\text{empty}) \in AM^*, \\
&\qquad\quad \nu \in AM,\ \omega \in AM^*
\end{aligned}$$

Using δ^*, we define the $AS(p)$ set. An abstract conversation sequence is considered to be possible over a given conversation protocol if and only if it is a sequence of abstract messages that begins from an initiating conversation state and ends in one of the terminating conversation states. Thus, given a conversation protocol p denoted by a tuple $(R, \Sigma, S, s_0, F, \delta)$, the $AS(p) \subseteq AM^*$ is defined as:

$$AS(p) = \{\omega \in AM^* \mid \delta^*(s_0, \omega) \in F\}$$

.

In the example protocol, there are four possible abstract conversation sequences (see Part I, Chapter 5, Section 5.2). Let us consider one of them: $< A, B, cfp >< B, A, propose >$ $< A, B,$*accept-proposal*$>$ $< B, A, inform >$. This sequence corresponds to $s_0 = A_1B_1 \rightarrow ... \rightarrow A_8B_8 \in F$ sequence of joint conversation states. In fact, it corresponds to the conversation described at the beginning of this chapter.

6.2 Conversation Systems

A conversation system is a set of conversations in a multi-agent system. In our model, a conversation system is denoted by a tuple (P, A, Λ, I, C). In this section, we describe these components in details.

Conversation Protocols (P): P is the set of conversation protocols of the conversation system, where each protocol is defined by a tuple as shown in Section 6.1.

Agents (A): A indicates the set of agents in the corresponding conversation system. Based on this parameter, we define another element in the model - 2^A. Using its formal definition–2^A is the set of all subsets of A–we refer to it as the set of all possible conversation groups in the conversation system. However, following the intuition that at least two agents must be involved in a conversation, we further restrict the definition of the 2^A set to be formulated as $2^A = \{g | g \subseteq A \text{ and } |g| \geq 2\}$.

Conversation Topics (Λ): Λ denotes the set of all conversation topics in the conversation system.

Intervals (I): An interval is a time period within the conversation system lifetime. Thus, we define I as follows: $I = \{[t1, t2] \mid t1, t2 \; time \; stamps \; , t_1 \geq 0, \; t2 \leq lifetime, \; t1 \leq t2\}$.

Conversations (C): A conversation in a conversation system is defined by a group of agents $g \in 2^A$ implementing a conversation protocol $p \in P$ on a conversation topic $\lambda \in \Lambda$ within a time interval $i \in I$ using an abstract conversation sequence $am^* \in AS(p)$. We can formulate the set of conversations, which is denoted as C, in a conversation system as follows:

$$C \subseteq \{(p, g, \lambda, i, m^*) \mid p \in P, \; g \in 2^A, \; \lambda \in \Lambda, \; i \in I, \; m^* \xleftarrow{g,\lambda,i} am^* \in AS(p)\}$$

Thus, a conversation $c \in C$ in a conversation system is a tuple (p, g, λ, i, m^*). The m^* parameter of a conversation denotes the actual conversation sequence that has taken place in the corresponding conversation. Thus, m^* is an implementation of some abstract conversation sequence over the corresponding conversation protocol. The actual conversation sequence m^* instantiates an abstract conversation sequence $am^* \in AS(p)$ with conversation group g, topic λ and time interval i. This instantiation is established as follows:

1. Instantiating conversation roles with agents: Here, we determine the mapping between conversation roles of the conversation protocol and the agent conversation group implementing it. For every conversation role $r \in R$, we determine an agent $a \in g$ ($g \in 2^A$) implementing it.

2. Instantiating abstract messages with a topic: Each abstract message of the implemented abstract conversation sequence is instantiated with the same topic, i.e. the topic of the conversation.

3. Instantiating abstract messages with time stamps: Finally, each abstract message of the implemented abstract conversation sequence is instantiated with a time stamp within a given time interval.

A conversation sequence m^* can therefore be denoted as $m^* = \mu_1 ... \mu_n$ where $\mu_i (\forall i, i = 1, .., n)$ denotes a message within the conversation sequence. A single message is defined as $\mu =< s, r, \sigma, \lambda, t >$, where s and r are the sender and the recipient of the message $(s, r \in g)$, σ is its message type, λ is its topic and t is its time stamp.

To demonstrate this formalization, we return to the conversation described at the beginning of this chapter. We denote the two conversing agents as $agent_x$ and $agent_y$, and their conversation group as $g = \{agent_x, agent_y\}$. Accordingly, $agent_x$ is the *initiator*, while $agent_y$ is the *participant*. We denote the topic of this conversation as $\lambda =$*contract-X* . Finally, we denote the interval of this conversation as $i = [t_1, t_4]$ assuming that the messages have been communicated at t_1, t_2, t_3, and t_4 time stamps. Thus, the actual conversation sequence of the given conversation can be represented as $<$ *agent*$_x$*, agent*$_y$*, cfp, contract-X,* t_1 $><$ *agent*$_y$*, agent*$_x$*, propose, contract-X,* t_2 $><$ *agent*$_x$*, agent*$_y$*, accept-proposal, contract-X,* t_3 $><$ *agent*$_y$*, agent*$_x$*, inform, contract-X,* t_4 $>$.

	Loss ($m < n$)	Insert ($m > n$)	Order ($m = n$)
Sequence Level	Loosing some messages of the actual sequence	Misoverhearing the actual sequence or misclassifying messages of another sequence	Inaccurately overhearing the order of messages in actual sequence
Message Level	Errors and Losses ($o_i \neq \mu_j$)		
	Missoverhearing or loosing some information of the overheard message (e.g. can not resolve the designated recipient of overheard message).		

Table 6.1: Possible differences between actual and overheard conversation sequences

6.3 Overhearing Conversations

An overhearing agent monitors inter-agent conversations by listening in to the exchanged communications. We denote the observed conversation sequence as o^* as opposed to m^*. The actual conversation sequence m^* is defined as $m^* = \mu_1 ... \mu_n$ where μ_i ($\forall i, i = 1, .., n$) denotes a message within the actual sequence. Analogously, we define the observed conversation sequence as $o^* = o_1 ... o_m$ in which o_i ($\forall i, i = 1, .., m$) denotes an observed message of o^*.

Since the overhearing agent may not overhear all messages, or may incorrectly overhear some messages, the overheard conversation sequence does not necessarily match the actual conversation sequence. Table 6.1 above summarizes the possible differences between the two conversation sequences.

Chapter 7

Algorithms for Conversation Recognition

Overhearing a conversation sequence o^*, one of the key objectives of the overhearing agent is to correctly recognize its appropriate conversation within the conversation system. Specifically, the agent should determine its conversation group (g), topic (λ), and interval (i). It must also identify the appropriate protocol (p) and its actual conversation sequence (m^*). We focus on the extraction of p and m^*, since extracting the other elements is almost trivial in many practical settings.

We propose a skeleton algorithm to determine the protocol corresponding to an observed sequence of messages o^* (Algorithm 2). Finding a matching protocol also enables us to determine its m^*.

The proposed skeleton algorithm follows similar principles to the debugging algorithm applied by [Poutakidis *et al.*, 2002]. The algorithm consists of three phases. Phase I is initialization (lines 1-2). Here, we construct a potential protocol set (PP) over P, which assumed to be given in advance. Each protocol in PP, called a control protocol, is an extension of the original protocol including a control mechanism used for performing phases II-III of the algorithm. At phase II (lines 4-13), we disqualify inappropriate protocols. For each observed message, each potential protocol is checked (line 10) using $CheckObsMsgMatch$. Inappropriate protocols are accumulated in the disqualified protocol set (DP) (line 12) and are subtracted from the PP set at the end of each iteration (line 13). Finally, at phase III (lines 15-16), we determine the final protocols, out of whatever protocols remain in the set PP.

This algorithm is a generic skeleton. Different instantiations are needed to handle the problems described in Table 6.1. Below, we first show an overhearing algorithm

Algorithm 2 FindMatchingProtocols
(**input**: observed sequence $o^* = o_1 o_2 ... o_m$,
output: protocol set $\subseteq P$)

```
1: // Phase I - Initialize
2: $PP = InitializePotentialProtocols(P)$
3:
4: // Phase II - Disqualify inappropriate protocols
5: for all $o_i$ in $o^*$ do
6:     if $PP$ is $empty$ then
7:         break
8:     $DP = empty\ set$
9:     for all $pp$ in $PP$ do
10:        $bool\ rc = CheckObsMsgMatch(o_i, pp)$
11:        if not $rc$ then
12:            $DP = DP \bigcup \{pp\}$
13:    $PP = PP \setminus DP$
14:
15: // Phase III - Determine final protocols
16: return $DetermineFinalProtocols(PP)$
```

for lossless o^* (Section 7.1). We remove this naive assumption, first in general lossy overhearing (Section 7.2), and then in systematic lossy overhearing (Section 7.3). Finally, Section 7.4 concludes by analyzing the complexity of those algorithms.

7.1 The Naive Algorithm

The Naive algorithm assumes that the observed conversation sequence is equal to the actual conversation sequence, i.e. it assumes no losses. In this case, *InitializePotentialProtocols* extends the original conversation protocols with two new components. The first is $s_{curr} \in S$–a pointer to the current conversation state within the protocol–it is initialized to s_0. The second is AG–a mapping between R and A–whose elements are initialized to unknown. We use the AG mapping to accumulate information about agents implementing various roles of the protocol.

Then, we check ($CheckObsMsgMatch$) whether exists a transition from s_{curr} to some s_{next} that is appropriate to the communicative act type of o. We also check whether agents, corresponding to this message, match the information in AG

($CheckRolesMatch$). In case these two conditions are satisfied, s_{curr} is incremented to s_{next} and procedure returns $true$, else it returns $false$.

Finally, each protocol, remaining in PP, is checked ($DetermineFinalProtocols$) to determine whether its $s_{curr} \in F$. If so, the corresponding protocol is considered as matching the observed conversation sequence.

7.2 The Random Loss Algorithm

The Random Loss algorithm handles the case in which there are multiple random message losses in o^*, where each such loss is made from up to k consecutive messages. This lossy overhearing condition may occur, for example, in case of malfunction in the overhearing agent, due to which it loses a certain interval within the overheard conversation.

In our example, in case $k = 2$, this algorithm can determine that $o^* =< agent_x, agent_y, \ cfp, contract\text{-}X, t_1 >< agent_y, agent_x, inform, contract\text{-}X, t_4 >$ corresponds to the FIPA protocol introduced at the beginning of Chapter 6. Furthermore, keeping track of the conversation state sequence within the protocol, it may be able to restore m^*.

In the Random Loss algorithm, control protocols are initialized with two additional components: CS and AG ($InitializePotentialProtocols$). The AG mapping has identical semantics as before. However, instead of a single s_{curr}, the CS set contains numerous pointers to the possible current conversation states reflecting the uncertainty caused by losing messages.

Procedure 3 CheckObsMsgMatch (of the Random Loss Algorithm)
(**input**: observed message $o = (sen, rcv, \sigma, \lambda, t)$,
control protocol $pp = (p, CS, AG)$ where $p = (R, \Sigma, S, s_0, F, \delta)$,
output: bool)

```
1: $NS = empty\ set$
2: for all $s_{curr}$ in $CS$ do
3:     $NS = NS \bigcup PropIgnLostMsg(s_{curr}, o, AG)$
4: $CS = NS$
5: return not ($CS$ is $empty$)
```

In $CheckObsMsgMatch$ (Procedure 3), for each s_{curr} in CS (lines 2-3), we determine its possible next states using $PropIgnLostMsg$. These next possible states are accumulated in NS set (line 3), which is then assigned to CS (line 4). If at the end of the procedure, CS is not empty, the procedure returns *true*, else it returns $false$.

Procedure 4 PropIgnLostMsg
(**input**: conversation state s_{curr},
observed message $o = (sen, rcv, \sigma, \lambda, t)$,
agent-role mapping AG,
output: conversation state set NS)

```
1: $NS = empty\ set$
2: $IS^0 = \{s_{curr}\}$
3: for $i = 0$ to $k$ do
4:    if $IS^i$ is empty then
5:       break
6:    $NS^i = IS^{i+1} = empty\ set$
7:    for all $s_{int}$ in $IS^i$ do
8:       $bool\ exists$ = check whether exists $\delta(s_{int}, < r_x, r_y, \sigma >) = s_{next}$
9:       if $exists$ and $CheckRolesMatch(o, < r_x, r_y, \sigma >, AG)$ then
10:         $NS^i = NS^i \bigcup \{s_{next}\}$
11:      $IS^{i+1} = IS^{i+1} \bigcup \{s | \delta(s_{int}, \_) = s\}$
12: $NS = \bigcup_{i=0}^{k} NS^i$
13: return $NS$
```

Given a s_{curr} state, $PropIgnLostMsg$ (Procedure 4) determines its next possible states ignoring up to k consecutive losses. In each iteration, we apply two sets–NS^i and IS^{i+1}. The first contains the next possible states corresponding to iteration i (line 10), whereas the second set holds up the intermediate states that are to be checked in the following iteration $i + 1$ (line 11).

Finally, we determine final protocols using procedure similar to the one shown in Procedure 4. A protocol is considered to be final if in its CS set there is at least one state which is either final or there is a final state with no more than k consecutive losses from it.

7.3 The Systematic Loss Algorithm

The Systematic Loss algorithm handles a more common situation in lossy overhearing–losing up to l conversation roles. This condition can occur in case that an overhearing agent, due to its location, cannot overhear messages sent from agents implementing the lost roles (e.g., the overhearing agent sees outgoing messages, but not incoming messages). In our example, in case $l = 1$ and the lost role is *initiator*, the algorithm can determine that $o^* =< agent_y, agent_x, propose, contract\text{-}X, t_2 >< agent_y, agent_x, inform, contract\text{-}X, t_4 >$ corresponds to the FIPA protocol described at the beginning of Chapter 6.

In the Systematic Loss algorithm, we determine for each set of lost roles (LR) a CS and AG component. Thus, for each potential protocol, we define a control set (CLR) that contains (LR, CS, AG) tuples.

Procedure 5 CheckObsMsgMatch (of the Systematic Loss Algorithm)

```
(input: observed message $o = (sen, rcv, \sigma, \lambda, t)$,
 control protocol $pp = (p, CLR)$ where $p = (R, \Sigma, S, s_0, F, \delta)$
      and $CLR = (\{(LR, CS, AG) | \forall LR \in LRS\})$,
 output: bool)
1: for all $(LR, Cs, AG)$ in $CLR$ do
2:    $NS = empty\ set$
3:    for all $s_{curr}$ in $CS$ do
4:       $bool\ exists$ = check whether exists $\delta(s_{curr}, < r_x, r_y, \sigma >) = s$
5:       if $exists$ and $CheckRolesMatch(o, < r_x, r_y, \sigma >, AG)$ then
6:          $NS = NS \bigcup PropIgnLostRoles(s, LR)$
7:    $CS = NS$
8:    if $CS$ is $empty$ then
9:       $CLR = CLR \setminus \{(LR, CS, AG)\}$
10: return not ($CLR$ is $empty$)
```

In $CheckObsMsgMatch$ (Procedure 5), each (LR, CS, AG) is considered individually (line 1). For each s_{curr} in its CS, we determine in lines 4-9 whether exists a potential next state and then propagate from it ignoring the lost roles (using *PropIgnLostRoles* in Procedure 6). The next potential states are accumulated in NS set, which is later assigned to CS (line 7). If CS is empty, the (LR, CS, AG) tuple is discarded from CLR (lines 8-9). The procedure returns *true* if at the end of it the CLR set is not empty, else it returns $false$ (line 10).

Procedure 6 PropIgnLostRoles
(**input**: conversation state s_{curr},
conversation role set $LR \subseteq R$,
output: conversation state set NS)

```
1:  $NS = empty\ set$
2:  $IS^i = \{s_{curr}\}$
3:  while $IS^i$ is not $empty$ do
4:      $IS^{i+1} = empty\ set$
5:      for all $s_{int}$ in $IS^i$ do
6:          for all $\delta(s_{int}, < r, \_, \_ >) = s$ do
7:              if $r \in LR$ then
8:                  $IS^{i+1} = IS^{i+1} \bigcup \{s\}$
9:              else
10:                 $NS = NS \bigcup \{s_{int}\}$
11:     $IS^i = IS^{i+1}$
12: return $NS$
```

7.4 Discussion

We now turn to analyzing the complexity of the conversation recognition algorithms we presented. The complexity of those algorithms is important to determine their appropriateness for large-scale settings since in such settings the overhearer is required to process large quantities of overheard messages.

The algorithmic skeleton (Algorithm 2) consists of three phases. However, only phases II and III contribute to algorithm complexity. In phase II, we match each of m messages with each protocol in PP using $CheckObsMsgMatch$. In phase III ($DetermineFinal\ Protocols$), we determine whether a protocol is final ($CheckIfProtocolFinal$). In this analysis, we denote the complexity of *Check-ObsMsgMatch* at iteration i as $O(f_1^i)$, while $CheckIfProtocolFinal$ procedure is denoted as $O(f_2)$.

The complexity of both phases depends on the number of protocols in PP at each stage of the algorithm. In the best case, all (but one) protocols are disqualified after the first iteration, whereas the final protocol remains through all m iterations. Assuming m is relatively big, the complexity of disqualifying $|P| - 1$ protocols is negligible. Thus, the best-case complexity can be formulated as follows:

$$(|P|-1)O(f_1^1)+\sum_{i=1}^{m}O(f_1^i)+O(f_2)=\sum_{i=1}^{m}O(f_1^i)+O(f_2)$$

In the worst case, all protocols remain consistent with all m messages, and are therefore repeatedly matched against overheard messages:

$$|P|(\sum_{i=1}^{m}O(f_1^i)+O(f_2))$$

Now, let us focus on evaluating the $\sum O(f_1^i)+O(f_2)$ component–the complexity for matching a single protocol–for each algorithm. We denote this complexity by $O(T)$. In the Naive algorithm, both $O(f_1^i)$ and $O(f_2)$ are equal to $O(1)$, since both procedures only perform a simple check. Thus, $O(T)=O(m)$ for the Naive algorithm. Thus, the Naive algorithm is efficient. However, its naive assumption, assuming no losses, is often challenged in real-world settings.

In the Random Loss algorithm, the complexity of $O(f_1^i)$ and $O(f_2)$ depends on the size of the appropriate CS set. The size of CS in iteration i is determined by the NS set of the previous iteration $(i-1)$. Furthermore, for each state in CS, we examine all states that are up to k transitions from it. Thus, in order to evaluate $O(T)$, we must consider the structure of function and the size of the NS set established in each iteration.

In the best case for $O(T)$, δ contains only one possible transition for each state and $|NS|$ is always equal to 1. Accordingly, $O(T)$ is equal to $mO(k)+O(1)=O(mk)$.

In the worst case, δ contains b transitions for each state–b is the branching factor of the state $(1\leq b\leq|\Sigma|)$. Thus, the complexity $O(\alpha)$ of examining up to k transitions from a certain state $s\in S$ can be evaluated as $O(1+b+b^2+...+b^k)\leq O(b^{k+1})$. Such states contribute no more than one new state to NS. In the worst case, $|NS|$ is $1+b+b^2+...+b^{k-1}\leq b^k$. We denote it as β. Thus, the complexity of $O(f_1^i)$ is $\alpha(1+\beta+\beta^2+...+\beta^m)\leq\alpha\beta^{m+1}=O(b^{mk+2k+1})$. Analogously, the complexity of $O(f_2)$ is $b^m(1+b+b^2+...+b^{n-m})=O(b^{n+1})$, where $n=|m^*|$. Thus, the worst-case complexity of $O(T)$ for the Random Loss algorithm is $O(b^{mk})+O(b^n)$.

In the Systematic Loss algorithm, the complexity of $O(f_1^i)$ and $O(f_2)$ depends on the size of CS and the structure of δ. However, this complexity also depends on the number of LR sets in CLR, i.e. $|CLR|$. In Section 7.3, we have defined each LR set as a possible combination of up to l lost roles of the protocols' conversation roles. Thus, $|CLR|$ can be formulated as follows:

$$|CLR| = \sum_{i=0}^{l} \left(\begin{array}{c} |R| \\ i \end{array} \right)$$

In the best case, $b = 1$ and $|NS|$ is always equal to 1. In addition, all LRs (but one) are disqualified after the first iteration. Furthermore, from each state, h states can be skipped ($PropIgnLostRoles$). Thus, the complexity of $O(f_1^i)$ is always $O(1 + h) = O(h)$, and the complexity of $O(f_2)$ is $O(1)$–simply checking the remaining state. Therefore, the best-case complexity of $O(T)$ for the Systematic Loss algorithm is equal to $O(mh)$.

In the worst case, we assume that no propagation can be made. Thus, the complexity of $O(f_1^i)$ is similar to the Naive algorithm, only multiplied by $|CLR|$, i.e. $|CLR|O(m)$. As for $O(f_2)$, from the single state in CS, all states in levels $n - m$ from it must be examined. Thus, similarly to the principles explained above, $O(f_2) \leq |CLR|O(b^{n-m+1}) = |CLR|O(b^{n-m})$. Thus, the worst-case complexity of $O(T)$ for the Systematic Loss algorithm is equal to $|CLR|(O(m) + O(b^{n-m}))$.

In general, it is difficult to determine which of the algorithms is better. However, in practice, we often know which roles are lost or at least know the number of lost roles. In such cases, the $|CLR|$ parameter becomes a constant and, thus, the Systematic Loss algorithm seems to be more efficient than the Random Loss algorithm.

Part III

Selective Overhearing

All previous investigations on overhearing (presented in Chapter 2) rely on the ability to overhear *all* relevant inter-agent communications. However, overhearing multi-agent settings, and in particular *large-scale* ones, this assumption can be challenged. In such settings, overhearing agent might not be able to overhear all the committed conversations due to limited resources. Instead, the overhearing agent must be *selective*, carefully choosing its targets.

Although both conversations and agents can be chosen as potential targets (both scenarios exist in the real-world), our work focuses on the latter. Thus, overhearing agent selectively targets communicating agents within the monitored system. Choosing one agent over the other, we assume that the values of information derived from overhearing each agent can be evaluated and compared in context of the task performed by the overhearer (e.g., monitoring progress, providing assistance and so on–see Chapter 2). Otherwise, the overhearing agent can simply decide on random targets.

We propose to use organizational knowledge on the monitored settings as a basis for this decision. Both human [Gannon and Newman, 2001, Best *et al.*, 2003] and multi-agent [Dignum, 2003, Horling and Lesser, 2004] organizations assume that their members differ in their roles, authority, etc. These differences also influence the characteristics of conversations committed by the agents (e.g., the number of conversations concurrently carried out by an agent, the significance of those conversations, etc).

In our work, we use this organizational knowledge to choose which agents should be overheard under the restriction of selectivity. We focus on hierarchical organizations and their specific characteristics. Hierarchical organizations are widely common both in the real-world settings (e.g., many corporates) and in the implementations of multi-agent systems (where hierarchical structures are mainly used for decomposition of complex tasks [So and Durfee, 1996, Yadgar *et al.*, 2003]).

In this part of the thesis, we perform an empirical research of selective overhearing in hierarchical organizations. Based on existing studies in various disciplines of social science [Dewan *et al.*, 1997, Gannon and Newman, 2001, Best *et al.*, 2003, Jensen, 2003, Friebel and Raith, 2004], we model and simulate inter-agent communications in hierarchical organizations (Chapter 8). Then, we empirically study overhearing of such organizations applying various centralized (Chapter 9) and distributed (Chapter 10) selective overhearing policies.

Chapter 8

Selective Overhearing of Hierarchical Organizations

This chapter takes the first steps towards an empirical study of selective overhearing in organizations. Focusing on hierarchical organizations, we first propose a model of selective overhearing in such organizations (Section 8.1). Then, in Section 8.2, we show how to instantiate this model simulating different types of hierarchical organizations and different types of selective overhearing strategies.

8.1 Modelling Selective Overhearing of Hierarchical Organizations

Overhearing extracts information from a *conversation system* (see Part II, Chapter 6, Section 6.2), the set of conversations generated by an organization. Conversation systems change based on the type of organization that is being overheard, and, in turn, overhearing agents must adapt their overhearing policies to match the conversation system. This section describes the conversation systems expected of hierarchical organizations (Section 8.1.1), describes a number of general selective overhearing policies for such organizations (8.1.2) and proposes a way to evaluate them (8.1.3).

8.1.1 Hierarchical Communication

Organizational communications have been the subject of continuous research in various disciplines of social science such as social behavior, organizational theory, information theory and strategic management [Best *et al.*, 2003, Dewan *et al.*, 1997, Friebel and Raith, 2004, Gannon and Newman, 2001, Jensen, 2003]. In these, communications in hierarchical organizations have been of a particular interest. We summarize the characteristics of hierarchical communications as follows:

- **Distribution Characteristic [Where do conversations take place?]**. No conclusive indication has been found, of a relation between the volume of an agent's communications and its hierarchy level [Jensen, 2003]. Thus, the *total* volume of conversations in a given hierarchy level depends on the number of agents associated with this hierarchy level. The latter, in turn, is determined by the hierarchical structure of the organization, i.e. the distribution of agents among the hierarchy levels. For instance, in common pyramidal hierarchies, the number of agents associated with each hierarchy level is smaller in higher hierarchy levels. Thus, most conversations in the pyramidal organizations are held between agents in the lower hierarchical levels, simply because most agents are associated with these levels.

- **Scope Characteristic [What do agents discuss?]**. Social science studies distinguish between three types of information: strategic, tactical and operational [Best *et al.*, 2003, Gannon and Newman, 2001]. These different types of information are associated with different organizational hierarchy levels. The top levels handle strategic information, the middle levels are responsible for the tactical information, and the lower levels handle operational information. Thus, in hierarchical organizations, agents communicate on information within their responsibility scope in the organization, i.e. mainly information associated with their hierarchy level or relatively close to it.

- **Span Characteristic [With whom agents communicate?]**. Formal communications in hierarchical organizations reflect the restricted flow of information in such organizations: either top-down or bottom-up [Dewan *et al.*, 1997, Friebel and Raith, 2004, Jensen, 2003]. Accordingly, agents communicate mostly with their peers, subordinates and their close superiors. Meaning that most communications are held between agents of the same hierarchy levels or between agents in relatively close hierarchical levels. However, formal communications alone fail to capture the rare (but occurring) communication between top-level and low-level agents. Such communications, that do not follow the strict hierarchical structure, are called informal communications [Jensen, 2003].

To formalize the characteristics presented above, we specialize and extend our model of conversation systems introduced in Part II, Chapter 6, Section 6.2. We define a conversation system of hierarchical organizations as a tuple (L, A, Λ, P, I, C), as explained below.

Hierarchy Levels **(L).** The set of hierarchy levels is an extension of the previous model. The hierarchy levels form the basis for the organizational information flow and roles.

Agents **(A).** A is the set of communicating agents within the organization, each associated with a hierarchy level. The distribution of agents among hierarchy levels determines the structure of the hierarchical organization. For instance, in pyramidal-hierarchies, the number of agents in higher hierarchal levels is always smaller than in the lower ones. In diamond hierarchies, the middle levels have more agents than bottom and top levels.

Conversation Topics **(Λ).** Λ denotes the set of conversation topics. Each topic is probabilistically associated with a corresponding hierarchy level such that the topics of a more strategic nature are associated with the higher end of the hierarchy and the more operational topics with the hierarchy's bottom (see Scope Characteristic above).

Conversation Protocols **(P).** P indicates the set of conversation protocols used in a conversation system (see Part II, Chapter 6, Section 6.1 for a detailed discussion). Intuitively, this is the set of conversation types that can occur, e.g., queries, brokering, informing, etc.

Intervals **(I).** An interval is a time period within the lifetime of an organization. Each conversation takes place over a certain interval within the I set.

Conversations **(C).** Based on the parameters above, we define a conversation as a group of agents $g \in 2^A$ implementing a conversation protocol $p \in P$ on a conversation topic $\lambda \in \Lambda$ within a time interval $i \in I$. Thus, the C set can be formulated as $C \subseteq \{(p, g, \lambda, i) | p \in P, g \in 2^A, \lambda \in \Lambda, i \in I\}$.

The characteristics of the C set depend on the type of organization. In hierarchical organizations, it follows the characteristics described at the beginning of this section. Specifically, these affect the selection of topics to be associated with each level, the assignment of agents to levels, the number of conversations at each level, and the selection of conversation partners in terms of their relative position in the organization. We accomplish these characteristics using a set of probability functions (see detailed discussion in Section 8.2.1).

8.1.2 Selective Overhearing Policies

This part of the thesis focuses on selective overhearing. We assume that an *overhearing policy* controls and coordinates multiple overhearing resources (for simplicity, we will refer to each as an overhearing agent). The policy assigns overhearing agents to conversations in the organizations. We distinguish two possible assignment types. In the first, the overhearing agent is assigned a single conversation chosen from all the conversations in the monitored organization. In the second, each overhearing agent can focus on a single communicating agent (referred as its *target*), overhearing all of its conversations. Both types of scenarios exist in the real-world.

We focus on agent-target assignments, where all conversations simultaneously carried out by the target are overheard, as long as the overhearing agent is currently listening to the target (i.e., only the conversations that take place within the overhearing time interval are overheard). During this time, the overhearing agent performs conversation recognition (see Part II, Chapter 7) for each conversation. Since conversation recognition takes time (to track and match the communicative acts being exchanged), the overhearing agent initially does not know the participants, protocol and topic associated with an overheard conversation. It must infer them from the contents of the messages it receives. The overhearing agent starts overhearing assuming that the conversation protocol and topic can be any of the $p \in P$ and $\lambda \in \Lambda$ respectively. Gradually, the overhearer is able to disqualify inappropriate protocols and topics until it determines the correct protocol and topic.

Assigning overhearing agents to their targets, we must take into account the limitation of *selectivity*. In large-scale multi-agent systems, overhearing resources are bound to be limited, and thus selected targets must be carefully chosen. Accordingly, only a subset of all potential targets can be covered. These targets are determined according to the applied overhearing policy. Different criteria can be used to determine the target agents to be overheard: For example, the targets can be chosen randomly, or the information at the different stages of the conversation recognition process can be used to determine whether to continue to overhear the current agent or to find another target. In this research, we propose to use the organizational knowledge on the monitored settings as a basis for this decision (e.g. agents' hierarchy levels, agents' activity level in the organization, etc). These organizational differences influence the characteristics of conversations carried out by the agents (see Section 8.1.1), and thus can be used as a promising basis for an effective assignment of overhearing agents to their targets.

8.1.3 Evaluating Overhearing Policies

To evaluate overhearing policies, we must first decide on an appropriate measure of the value of information derived from overhearing. After all, overhearing is a monitoring approach: It aims to provide information about a monitored system (here, an organization, an MAS). The problem is that different monitoring tasks have different information needs, and therefore the value of monitored information may be qualitatively different from one monitoring task to another. For instance, to visualize the state of an organization with respect to a given task, monitoring emphasizes completeness, where as missing details are those most sought after, regardless of their importance in the organization [Kaminka *et al.*, 2002]. On the other hand, if the monitoring task is to determine information about a given topic, then only conversations on this topic will be deemed valuable. Due to the nature of hierarchical conversations (as discussed earlier), these may be clustered around specific levels of the organization.

To allow us unbiased treatment of the value of conversations–to the degree possible–we assume that the value of overheard conversations is a function of its value to the organization, and will thus utilize the organizational value as the basis for evaluating overhearing policies. This simplifying assumption implies that we assume targets are truthful and do not modify their conversations because they are overheard. It also implies that we are actually assuming that the organizational information value is proportional to the overhearing information value, at least to the degree that we interpret lower information value as worse. The same assumption is also made by most previous overhearing applications described in Chapter 2, Section 2.1.

Expressing the value of overheard information extracted from overhearing agent $a \in A$ involved in a conversation $c \in C$, we consider two type of parameters. The first addresses the conversation itself–e.g. its topic, content, context, etc. We denote it as ρ^c. The second type of parameters, denoted as ρ^a, addresses the agent being overheard: its role in the conversation, its position in the organization, etc. Thus, the value of overhearing agent $a \in A$ involved in conversation $c \in C$ is a function of both these parameter types and can be formalized as $\nu(c, a) = \psi(\rho_1^c, \rho_2^c, ..., \rho_1^a, \rho_2^a, ...)$.

In hierarchical organizations, we expect the more valuable conversations to be held in the higher hierarchical levels. As discussed in Section 8.1.1, conversations at the top of the hierarchy are more strategic, whereas the more operational conversations are held at the lower levels [Best *et al.*, 2003, Gannon and Newman, 2001].

8.2 Simulating Selective Overhearing of Hierarchical Organizations

This section presents an instantiation of the model, shown in the previous section, to simulate selective overhearing of hierarchical organizations. We show how different types of hierarchical organizations can be simulated by controlling various parameters, such as the shape of the organization (diamond, pyramidal, inverse-pyramidal, etc.), the number of layers it contains, and the value of information of its conversations.

The number of communicating agents, i.e. $|A|$, was set to 50 simulating relatively small organizations. The number of hierarchy levels, i.e. $|L|$, was initially set to 7. However, changing this value as done in Chapter 9, Section 9.2.2, we can control the hierarchy height of the monitored hierarchical organization.

Our research mainly focuses on pyramidal hierarchies–the most common type of hierarchical organizations. To simulate a pyramidal organization, agents were distributed between different hierarchy levels according to a Zipf-like hyperbolic distribution. The probability of an agent to be associated with hierarchy level l, $1 \leq l \leq |L|$ was set to $1/l$ (normalized). Accordingly, the number of agents assigned to each hierarchy level becomes smaller as the hierarchy levels get higher. Although focusing on pyramidal hierarchies, we also examine other shapes of hierarchical organizations such as diamond, inverse-pyramidal and etc (see Chapter 9, Section 9.2.3). These other organizational structures can be simulated by simply changing the probability function determining the distribution of agents between different hierarchy levels.

The number of topics, i.e. $|\Lambda|$, has been set to 80. This value reflects our intuition that each agent has at least one conversation topic unique to itself (under its direct responsibility) and a few more that are shared with its peers. The number of protocols was defined as 25 simulating a diversity of interactions that are possible in organization. The duration of each protocol, defining the length of a conversation, was randomly set for each protocol to a value within {5,10,15,20,25}. We denote it as $d(p)$.

The number of overhearing agents was initially set to 15. Thus, the overhearing coverage, defined as the ratio between the number of overhearers and the number of communicating agents—$k/|A|$, was set to 30%. This overhearing coverage ratio corresponds to the assumption of limited overhearing resources, i.e. not being able to cover all communicating targets. In later experiments, we change the number of overhearing agents to examine the affect of coverage on overhearing results.

The value of overheard information, $\nu(c, a)$, was evaluated using two parameters: one of the ρ^c type and the other of the ρ^a type. As a ρ^c parameter, we considered the conversation's topic. Each conversation topic $\lambda \in \Lambda$ has been randomly given a value $\nu(\lambda)$ between 1 and 100. In addition, for each hierarchy level we defined a value range of its conversation topics. This value range was calculated as a relative portion of [1,100] equally divided between the levels such that the higher hierarchies have the greater values. This way, the randomly set $\nu(\lambda)$ also associates the topic $\lambda \in \Lambda$ with its hierarchy level.

As the second parameter, of the ρ^a type, we considered the agent's role in the overheard conversation. For each conversation protocol $p \in P$ two roles, denoted as $r \in R(p)$, have been defined. Their values were randomly set to one of the following combinations: {50,50}, {67,33}, {75,25} and {99,1}. In this manner, we simulate differences in overhearing different roles within the same conversation.

Finally, the value of overheard conversation has been calculated using an accumulative function. The value of overhearing agent $a \in A$ in a conversation $c \in C$ is defined as a sum of the value of the conversation's topic and the value of agent's role in the conversation protocol, i.e. $\nu(c, a) = \nu(\lambda) + \nu(r)$. Thus, the value of overheard conversation ranges from 2 to 199.

8.2.1 Generating conversations

In the experiments in Chapters 9 and 10, we generated conversation systems and simulated their dynamic execution, in a manner consistent with the characteristics of hierarchical organizations, described earlier in Section 8.1.1. At the beginning of each simulation run, $|C_t|$–the number of conversations at time t–new conversations are generated using the procedure below. Then, each time a simulated conversation ends, a new conversation is generated. Thus, a constant *level of conversation activity* is maintained throughout the $lifetime$ of the conversation system (fixed at 1000).

The procedure for generating a single conversation at time t followed the steps shown in Algorithm 7. We assume that each simulated conversation involves two communicating agents. However, this procedure can easily be extended to support larger conversation groups.

Algorithm 7 Simulate Conversation Generation at Time t

```
1: $l_1 \leftarrow$ choose hierarchy using $Pr(L)$
2: $a_1 \leftarrow$ choose agent of hierarchy $l_1$
3: $\lambda \leftarrow$ choose topic given hierarchy $l_1$ using $Pr(\Lambda|L)$
4: $l_2 \leftarrow$ choose hierarchy given topic $\lambda$ using $Pr(L|\Lambda)$
5: $a_2 \leftarrow$ choose agent of hierarchy $l_2$
6: $p \leftarrow$ choose protocol using $Pr(P)$
7: set $a_1$ and $a_2$ to implement roles in $R(p)$
8: return $c = (p, \{a_1, a_2\}, \lambda, [t, t + d(p)])$
```

First in lines 1-2, we choose a level l_1 according to the distribution determining the organizational structure–a Zipf-like distribution for pyramidal hierarchies. We then arbitrarily select an agent a_1, associated with l_1, to initiate the conversation. The use of $Pr(L)$ ensures that the Distribution Characteristic, presented in Section 8.1.1, holds.

Next (line 3), a conversation topic λ is chosen using the conditional probability $Pr(\lambda|l_1)$, calculated according to Bayes' rule as $[Pr(l_1|\lambda) \cdot Pr(\lambda)]/Pr(l_1)$. $Pr(l_1)$ is known from the distribution determining the organizational structure. $Pr(\lambda)$–the probability that the conversation is carried out on topic λ–is assumed to be taken from the uniform distribution over Λ.

The calculation of $Pr(l_1|\lambda)$ requires some explanation. We remind the reader that a topic λ has a value v in the range [1,100], which determines its associated hierarchy level. We define a normal distribution with mean $\mu = v$ and standard deviation $\sigma = 0.5$. The value $Pr(l_1|\lambda)$ is given by this distribution. Intuitively, this translates into ensuring that agents usually carry out conversations on topics associated with their hierarchy level or relatively close to it–Scope Characteristic.

The next step is to determine the level of the other agent (lines 4-5). Here the process is reversed. We sample the topic's normal distribution to determine a new overhearing value, and its associated level l_2. We again arbitrarily select an agent a_2 associated with this level. Thus, $a_1$and a_2 are likely to be associated with the same hierarchy level, or close, thus, following the Span Characteristic.

Finally, a conversation protocol p is randomly chosen from the uniform distribution over P (line 6). We assign roles to the two agents from $R(p)$ (line 7). To reflect the intuition that agents of higher hierarchies commit the more valuable conversations, according to the Scope Characteristic in Section 8.1.1, we constrain the assignments such that 80% of assignments give the agent of higher hierarchy, the role of higher value.

8.2.2 Comparing Overhearing Policies

Given a set of generated simulated conversations (a conversation system), we can now begin to apply different overhearing policies to study their characteristics. To do this, we utilize the conversation value as described above. Given a policy *pol*, we use Algorithm 8 to calculate pol_{val}, *pol*'s *overhearing value* for the same conversation system. pol_{val} is the accumulated value of all conversations overheard using *pol*.

Algorithm 8 Calculate Policy Overhearing Value

```
1: $pol_{val} \leftarrow 0$
2: for all $t$ such that $0 \leq t \leq lifetime$ do
3:     $OC_t \leftarrow$ overheard conversations at time $t$
4:     $OA_t \leftarrow$ $k$ overheard agents at time $t$
5:     for all $c = (p, g, \lambda, i)$ such that $c \in OC_t$ do
6:         $pol_{val} \leftarrow pol_{val} + \nu_t(c, a) \quad \forall a, a \in g \ \wedge a \in OA_t$
7: return $pol_{val}$
```

Algorithm 8 presents the calculation of overhearing value for a team of k overhearers implementing specific policy *pol*. For each time unit t (lines 2–6), we calculate its overhearing value at time t and accumulate it in pol_{val} (line 6). An overhearing value at time t is defined as an accumulative conversation value of overheard agents. Thus, in lines 3–6 , for each overheard conversation, in a set of overheard conversations at time t (OC_t), its conversation value is accumulated for each communicating agent that has been overheard (the OA_t parameter indicates the set of agents overheard at time t).

However, calculating the value pol_{val} is insufficient. Conversation systems may differ due to factors such as random selection of topics and values, and our controlling of various parameters. Thus, it would be impossible to compare the absolute pol_{val} derived from a conversation system A, to that derived from a different conversation system B.

To overcome this difficulty, we normalize pol_{val} based on the value of the theoretical optimal value that could be derived for a given conversation system. Given k overhearing agents and a generated conversation system, we use Algorithm 9 to calculate the optimal overhearing value (*optimum*). *optimum* is a theoretical value, calculated based on full knowledge of the true value of all conversations in the organization: With each clock cycle, the k best targets are selected and assigned to overhearing agents (remember that k is relatively smaller than the number of potential targets due to selectivity restriction). We compare different policies by computing their value as $\frac{pol_{val}}{optimum}\%$.

Algorithm 9 Calculate Optimal Overhearing Value

```
1: $optimum \leftarrow 0$
2: for all $t$ such that $0 \leq t \leq lifetime$ do
3:     $\nu_t(a) \leftarrow 0 \quad \forall a,\ a \in A$
4:     for all $c = (p, g, \lambda, i)$ such that $c \in C_t$ do
5:         $\nu_t(a) \leftarrow \nu_t(a) + \nu_t(c, a) \quad \forall a, a \in g$
6:     $A_{t,k} \leftarrow k$ agents in $A$ with highest $\nu_t(a)$ values
7:     $optimum \leftarrow optimum + \nu_t(a) \quad \forall a, a \in A_{t,k}$
8: return $optimum$
```

Algorithm 9 introduces the calculation of *optimum*. For each time unit t (lines 2–8), optimum at time t is calculated and accumulated in *optimum* (line 7). The optimum at time t for k overhearing agents is defined as a sum of conversation values of k agents with the highest conversation values at time t (lines 6–7). A conversation value of agent $a \in A$ at time t–denoted as $\nu_t(a)$–is the accumulative value of its conversations at time t (lines 4–5). This algorithm makes a simplifying assumption on changing overhearing targets. It assumes that a change of overhearing target by an overhearing agent is instantaneous and has no cost. This assumption is also used in other calculations.

Chapter 9

Empirical Study of Centralized Policies

This chapter presents an empirical study of centralized selective overhearing policies in hierarchical organizations. Each overhearing policy may choose to overhear different target agents, and thus overhears different conversations. Consequently, some policies may perform well while others perform poorly. Furthermore, the same overhearing policy may vary in its performance, in principle, under different configurations of conversation systems and overhearing resource constraints. The experiments we report on seek to determine the qualitative performance of the policies and the factors that influence their performance.

We now compare several overhearing policies using their evaluation values (as a percentage of optimum–see Chapter 8, Section 8.2.2) in different configurations of hierarchical organizations. Each evaluation is performed based on an average of 50 independent experiments with the same parameters. Thus, in the figures below, each data point corresponds to 50 trials.

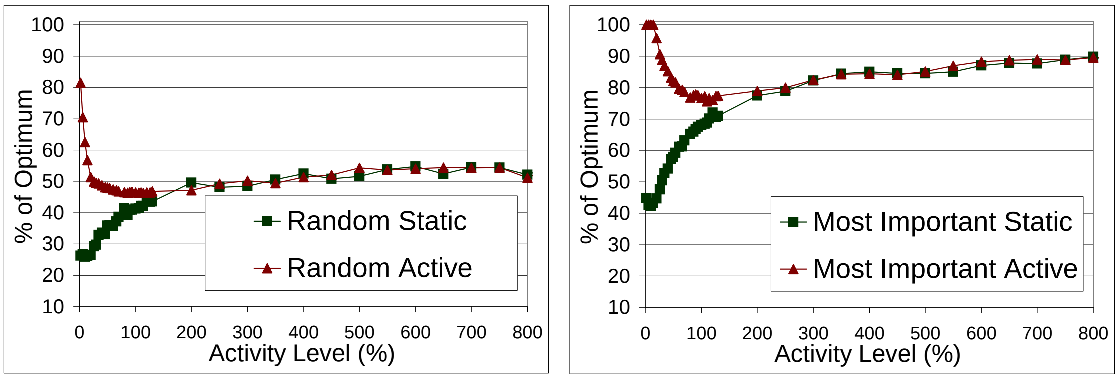
9.1 Static vs. Active Policies

Our initial hypothesis has been that the most successful overhearing in pyramidal-hierarchical organizations (under the restriction of selectivity) would be achieved by overhearing conversations of the most important agents–agents of highest hierarchical levels. The main intuition behind this hypothesis is that most important agents carry out the most valuable conversations. We refer to this type of policy as *Largest Value*.

Several largest-value policies are possible. In our first such overhearing policy, called *MostImportantStatic*, k overhearing agents were set to overhear the k most important agents (in terms of their hierarchy level). So as to never miss a conversation carried out by these agents, the policy committed to monitoring them regardless of whether they are currently communicating or not.

To evaluate this policy, we define baseline overhearing policy, called *RandomStatic*. Here, k overhearers were set to target k random agents chosen at the beginning of the experiment. Just as *MostImportantStatic* does not switch targets, neither does *RandomStatic*.

A potential drawback of these policies is that their overhearing targets are determined statically. In cases where the overheard agent is idle, overhearing it has zero value. We thus contrast these static policies with *active* policies, in which the selection of targets is made out of those agents that are communicating at the moment of selection (though the agent may not know at what stage in the conversation they may be). The *RandomActive* chooses k target agents, similarly to *RandomStatic*. However, each time a target is idle, an alternative target is randomly chosen. The *MostImportantActive* policy improves on *MostImportantStatic* by choosing the k most important agents from those that are currently active.



(a) Random (b) MostImportant

Figure 9.1: Static vs. Active Policies

Figures 9.1-a,b compare these policies. The values on the X-axis show the activity levels of the examined conversation systems, i.e. the ratio between the number of conversations at time t ($|C_t|$) and the number of communicating agents ($|A|$) (note that each agent may engage in more than one conversation in parallel). The Y-axis measures performance as percentage of the optimum. A comparison between Figure 9.1-a and 9.1-b shows that the two *Largest Value* policies outperform the random policies in most activity levels.

However, more importantly, there is a qualitative difference in the behavior of the active and static policies. In low activity levels, the likelihood of a given agent being idle is relatively high. In such settings, active policies outperform static policies. However, as the activity level grows, the probability of an agent to be idle reduces. Thus, static overhearing policies monotonically rise as the activity level grows until the probability of an agent to be idle is close (or equal) to 0. Indeed, in high activity settings, the difference between static and active policies is insignificant.

Nevertheless, active policies may not always be preferable, since they require qualitatively different knowledge about the monitored organization. Active policies rely on the ability to detect agents that are conversing at any given time, unlike static policies. Thus, a trade-off exists between the need to improve the results of overhearing, and the additional costs that may be required in detecting activity of potential targets.

9.2 Value vs. Volume Policies

As moving from static to active policies increased the overall volume of overheard conversations (and thus the total derived value), a second overhearing approach–*Largest Volume*–suggests itself. We implemented a version of it, called *MostActive*, which targets the k most active agents, i.e. the k agents that are carrying out the highest number of conversations at time t. Since the overhearing agent overhears all conversations committed by its target, the idea is that this policy will be more productive due to the greater quantity of overheard conversations.

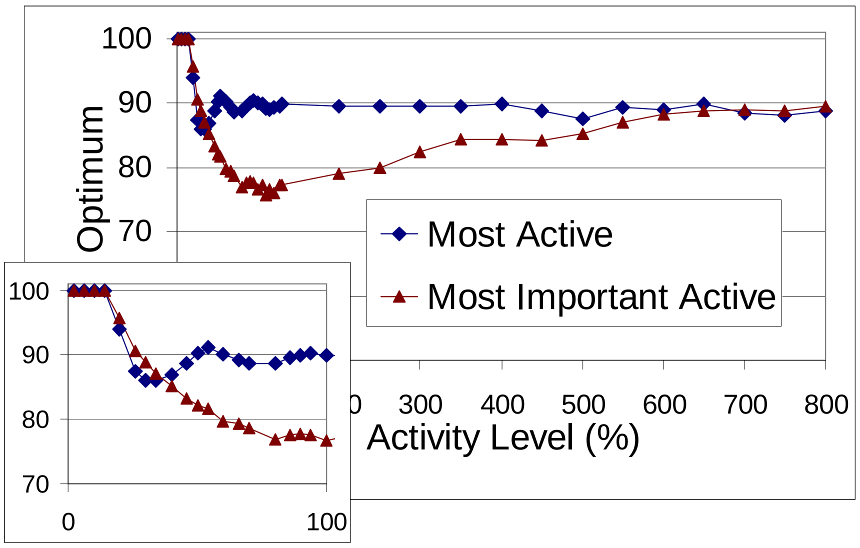


Figure 9.2: Value vs. Volume Policies

Figure 9.2 shows the performance of the *MostActive* and *MostImportantActive* policies. It shows that overhearing volume can in fact be a successful policy, using less knowledge about the monitored organization. The *MostActive* policy does not require knowledge of the organizational role of the targets. This result is surprising given that in pyramidal-hierarchical organizations, most conversations are held between agents of lower hierarchy levels. Thus, in fact, *MostActive* statistically targets the less important agents.

Furthermore, Figure 9.2 indicates that a tradeoff between the two types of policies exists. While the *Largest-Value* policies focus on overhearing a small number of highly valuable conversations, the *Largest-Volume* policies concentrate on overhearing a great volume of less valuable conversations. We dedicate the remaining part of this section to exploring the factors influencing this tradeoff.

9.2.1 The Overhearing Coverage

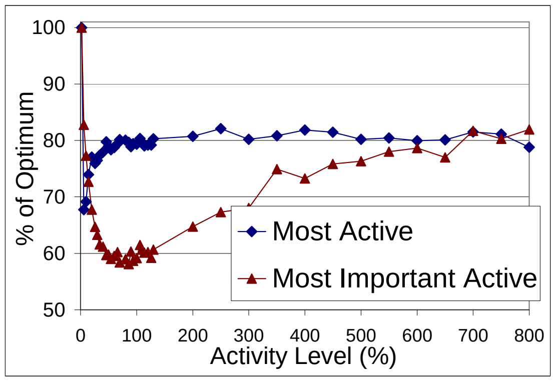
We begin by analyzing the performance of the two policies under conditions of different overhearing coverage–the ratio of the number of overhearing agents to the number of potential targets. These results are shown in Figures 9.3-a to 9.3-f. It can be seen that both policies become more efficient with higher overhearing coverage.

Clearly, this conclusion is to some extent straightforward. However, an additional, less-trivial conclusion is that relative performance of these policies does not change with selectivity. While increased coverage (reduced selectivity) increases the performance of both policies, the *MostActive* policy remains on top. It can be seen that the parabolic curve of *MostImportantActive* graph becomes less pronounced. In large overhearing groups, this effect can be explained by a significant overlap in overhearing targets for both policies.

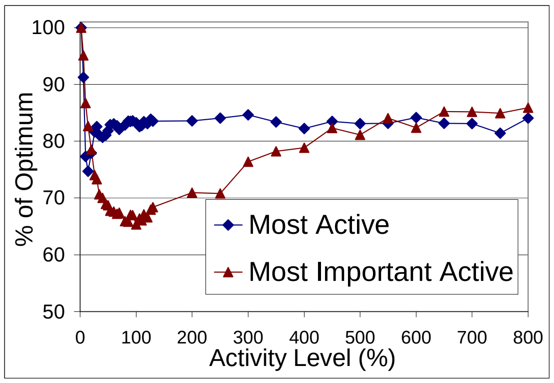
9.2.2 The Height of the Hierarchy

We now turn to analyzing the effect of hierarchies on the performance of the proposed policies. Figures 9.4-a to 9.4-e show the policies' behavior in organizations with 3, 5, 7, 9 and 11 hierarchy levels. The figures show that no significant change occurs in the performance of both policies. Instead, only a slight performance decrease occurs when the number of hierarchy levels is larger.

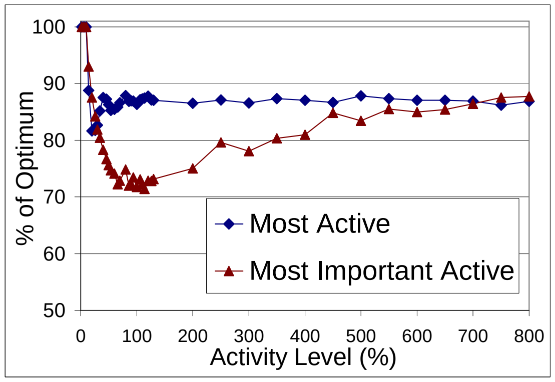
This lack of change is caused by the two policies essentially marking two extremes in the space of policies in overhearing hierarchical organizations; they tend to prefer the top and bottom levels. The *MostImportantActive* policy tends to always prefer



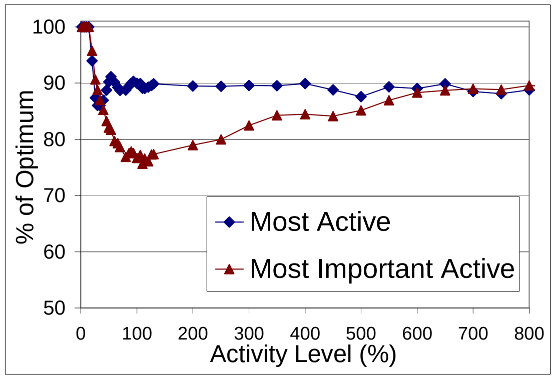
(a) Coverage: 5%



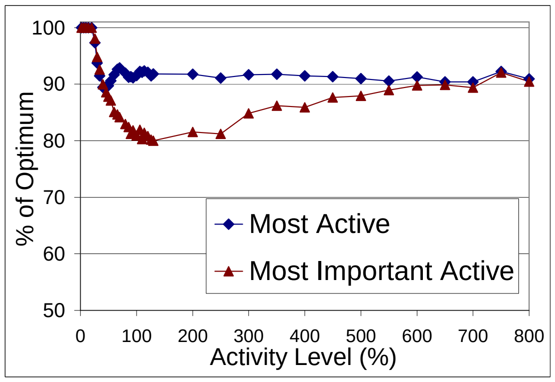
(b) Coverage: 10%



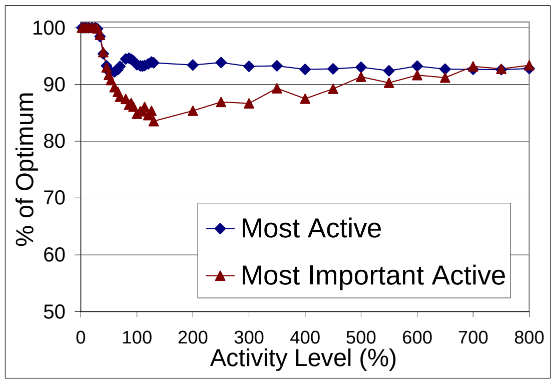
(c) Coverage: 20%



(d) Coverage: 30%



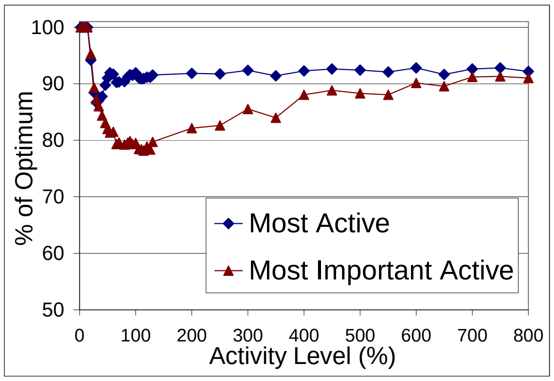
(e) Coverage: 40%



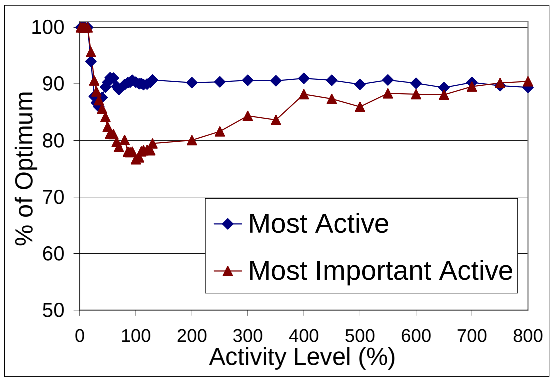
(f) Coverage: 50%

Figure 9.3: Effect of Coverage

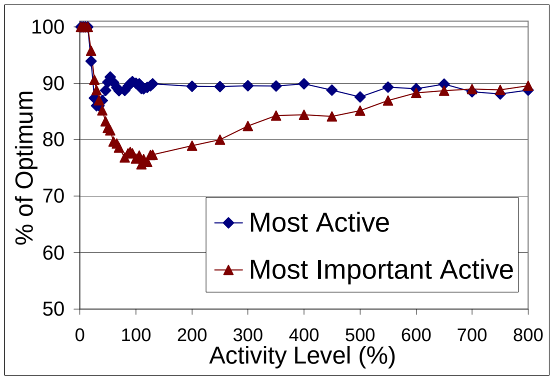
agents in the top level. The *MostActive* policy tends to prefer the bottom level (where there is more activity). Thus the middle levels in the organizations tend to be ignored by these policies, regardless of the number of such middle levels.



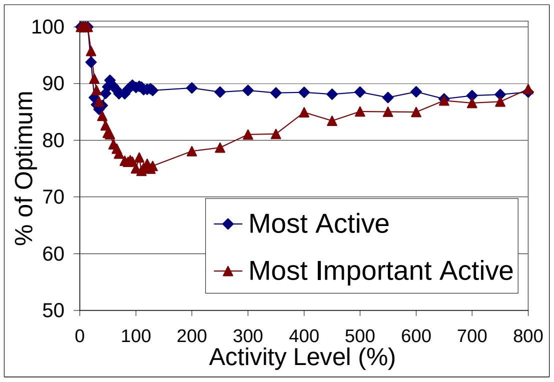
(a) 3 Hierarchy Levels



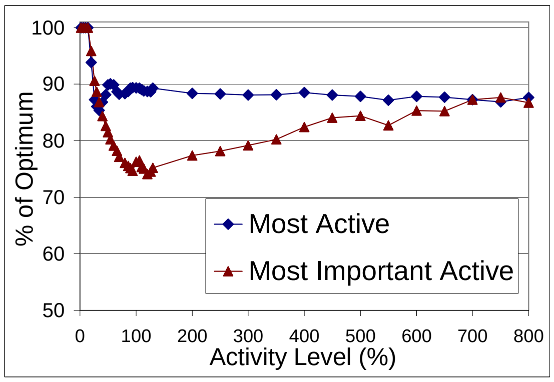
(b) 5 Hierarchy Levels



(c) 7 Hierarchy Levels



(d) 9 Hierarchy Levels



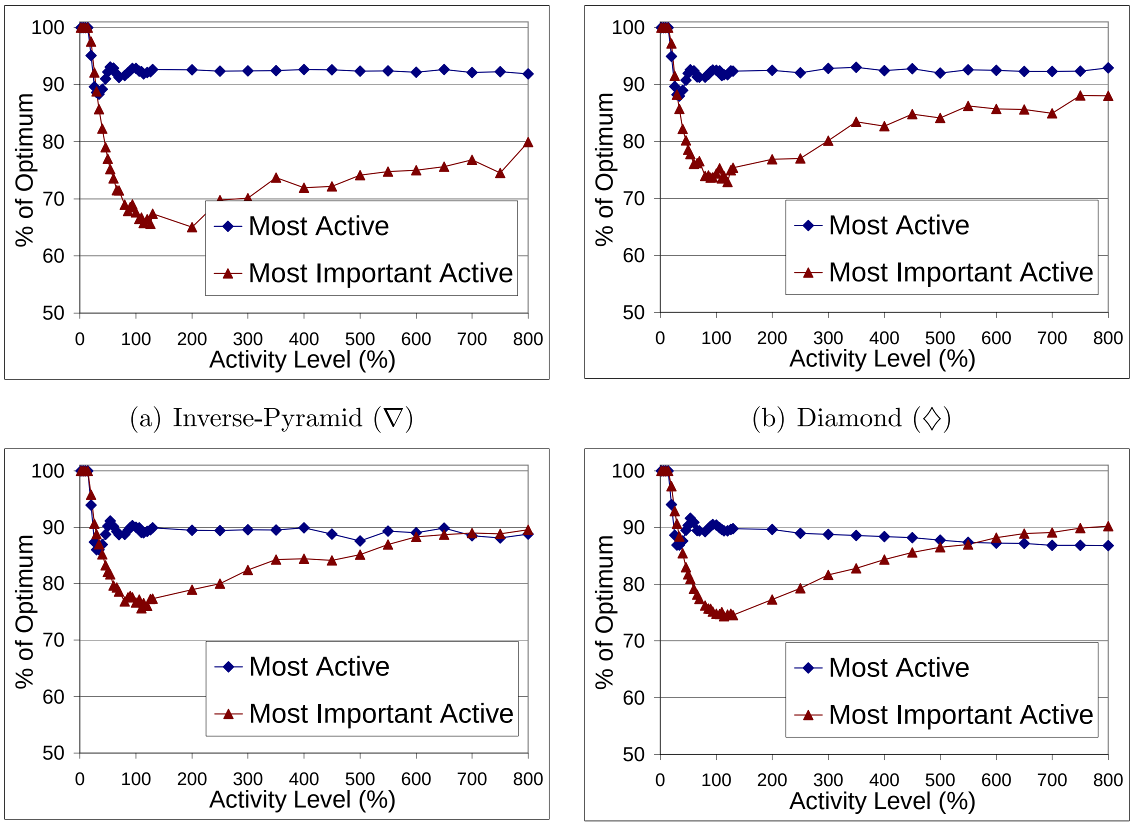
(e) 11 Hierarchy Levels

Figure 9.4: Effect of Hierarchy Levels

9.2.3 The Organizational Structure.

The next parameter we seek to explore is the organizational structure of hierarchical organizations. Figures 9.2, 9.3 and 9.4 above have shown the performance of the two policies in pyramidal hierarchical organizations. In this section, we study the behavior of the *MostImportantActive* and the *MostActive* policies in hierarchical organizations with other (than pyramidal) organizational structures.

Organizational structure of hierarchical organization is determined by the distribution of agents among hierarchy levels. In this work, we address three types of organizational structures besides pyramidal hierarchies. The first organization structure, we consider, is an *Inverse-Pyramid*. In such organizational structure, the number of agents in each hierarchy level becomes bigger as the hierarchy levels get higher. The second has a *Diamond* organizational structure. Here, the middle hierarchy level has the highest number of agents. Besides, as the hierarchy levels get either higher or lower, the number of agents associated with these levels becomes smaller. The last organizational structure, i.e. *Equally-Distributed*, is the one where the number of agents associated with each hierarchy level is more or less equal.



(a) Inverse-Pyramid (∇)

(b) Diamond ($\diamondsuit$)

(c) Pyramidal ($\triangle$)

(d) Equally-Distributed ($\square$)

Figure 9.5: Effect of Organizational Structure

The results of *MostImportantActive* and *MostActive* policies in these organizational structures (together with the pyramidal structure) are shown in Figures 9.5-a to 9.5-d respectively. Again, it can be seen that the relative performance of the two policies remains unchanged in different organizational structures. The main difference between these organizational structures is the rate of the convergence, i.e. the activity level at which the two policies intersect/meet for the second time. We intentionally cut the Figures 9.5-a,b,c and d at activity level of 800% so to emphasize the difference in gaps between the *MostImportantActive* and *MostActive* policies at this activity level. Thus, we can see that the slowest convergence is found in *Inverse-Pyramids*, then in *Diamond* structures, then come the *Pyramidal* hierarchies and, finally, *Equally-Distributed* structures show the quickest convergence.

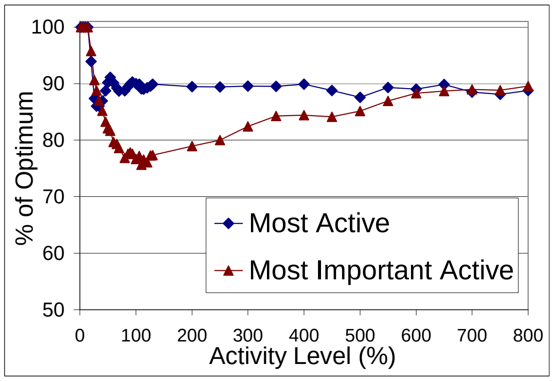
9.2.4 The Importance of Importance

It would seem that the relative performance of the two policies, is qualitatively unaffected by the selectivity level, nor by the height of the hierarchy (measured in number of levels) and neither by the different organizational structures. Yet hierarchical organizations are not characterized solely by these parameters. Rather, it is the difference in the importance of the different levels that is significant.

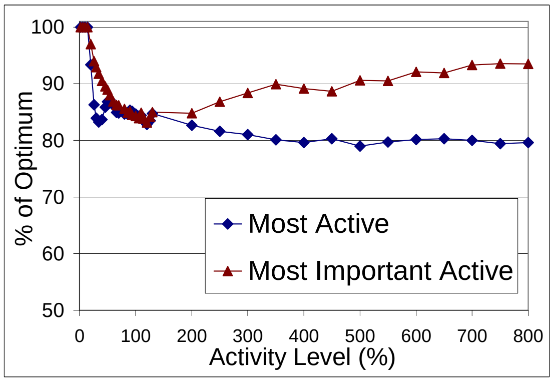
In a final set of experiments addressing the exploration of the value-volume tradeoff, we changed the *importance ratio* between the low-value and the high-value conversations, i.e. the ratio between the average values of conversations in the bottom and top hierarchy levels, respectively. In the previous experiments, the value of conversations ranged from 2 to 199. On average, conversations committed by agents of lowest hierarchy level were valued close to 50, while conversations of highest-level agents were valued around 150 (ratio of 1:3). In these experiments, we examine the two policies with additional ratios.

Figures 9.6-a,b,c show the performance of *MostActive* and *MostImportantActive* for importance ratios of 1:3, 1:5 and 1:8 ratios. It can clearly be seen that as the ratio of conversations value increases, the *MostImportantActive* policy improves (in all activity levels), while the *MostActive* policy deteriorates. At some point (Figure 9.6-b), the two policies shift relative places, and the *MostImportantActive* policy dominates.

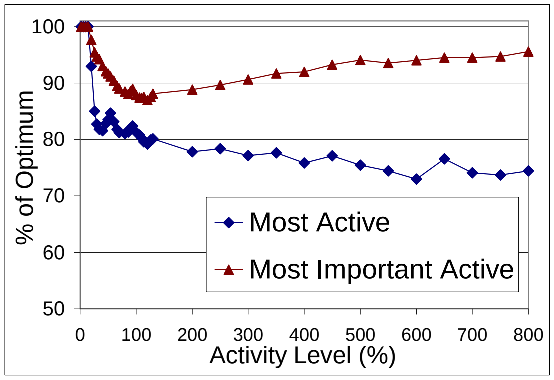
Thus, in case the ratio of high-level and low-level conversation values is significant, it is better to target highly important agents than to overhear low-level, highly-communicative ones. For intermediate ratios, a tradeoff exists between these policies, and for high ratio values, no tradeoff exists.



(a) Conversations Value Ratio 1:3



(b) Conversations Value Ratio 1:5



(c) Conversations Value Ratio 1:8

Figure 9.6: Effect of Conversations Value Ratio

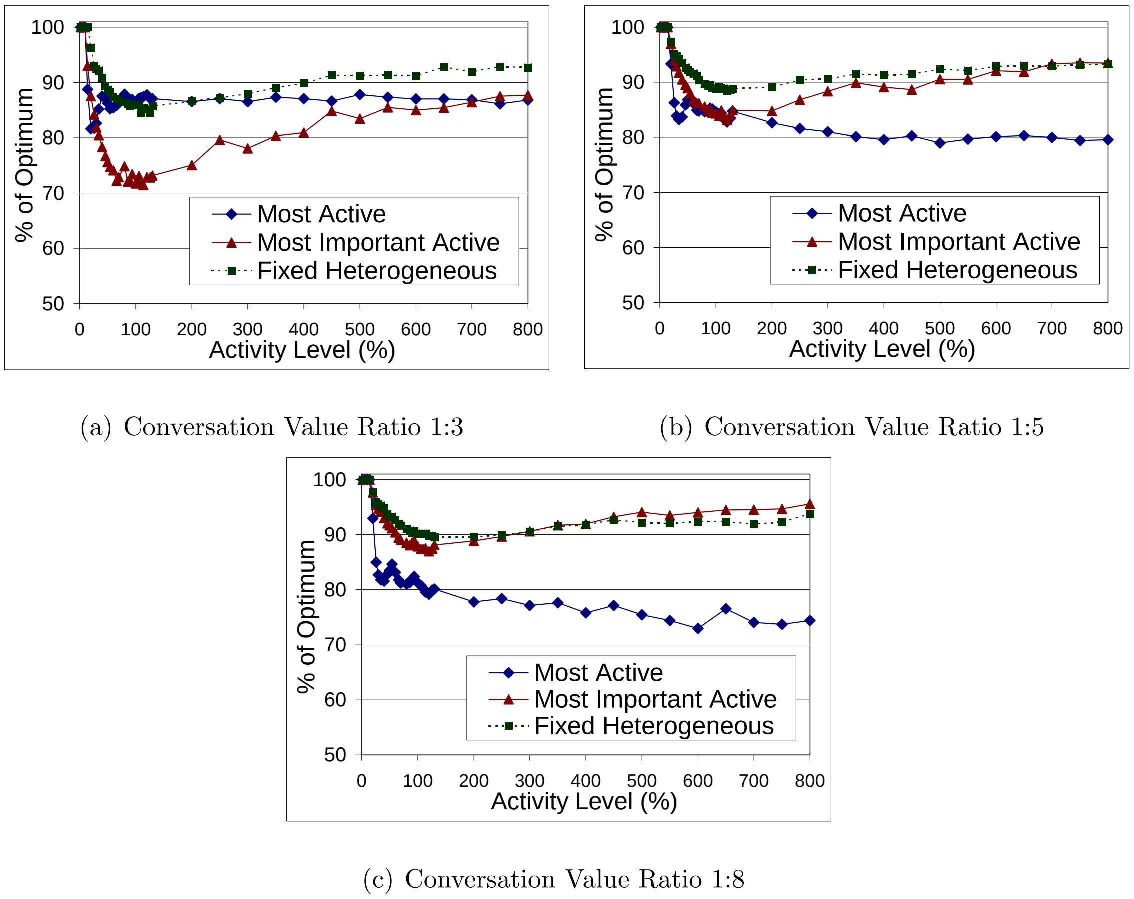
9.3 Heterogeneous Policies

Based on the insight that a classical value-volume tradeoff exists in centralized selective overhearing, we now tackle the question of how to apply these two policies effectively. An overhearing agent may not know in advance the exact parameters of monitored organization such as its activity level, conversation value ratio between high and low hierarchy levels, etc. Thus, it would face a dilemma deciding what policy is best to apply.

In this section, we consider a combination of the two policies in what we call *heterogeneous* selective overhearing. Here, instead of a single homogeneous overhearing group following a ceratin overhearing policy, we consider a heterogeneous overhearing group where some agents follow the *MostImportantActive* policy and others use the *MostActive* policy.

9.3.1 Fixed Heterogeneous Policies

A fixed division of an overhearing group between the two policies was first to be considered. Figures 9.7, 9.8 and 9.9 show the performance of heterogeneous overhearing under conditions of different fixed divisions. In Figures 9.7-a,b and c, 75% of overhearing agents use the *MostImportantActive* policy (Value), while the other 25% apply the *MostActive* policy (Volume). Figures 9.8-a,b and c show the performance of heterogeneous policies where the overhearing group is divided 50%-50% between the *MostImportantActive* and the *MostActive* policies. Finally, Figures 9.9-a,b and c show the results for a combination of 25%-75%. Each of the Figures 9.7, 9.8 and 9.9 show the performance of heterogeneous policies (with an appropriate fixed division) under the conditions of different conversation ratios compared to the performance of the *MostImportantActive* and the *MostActive* homogeneous policies.

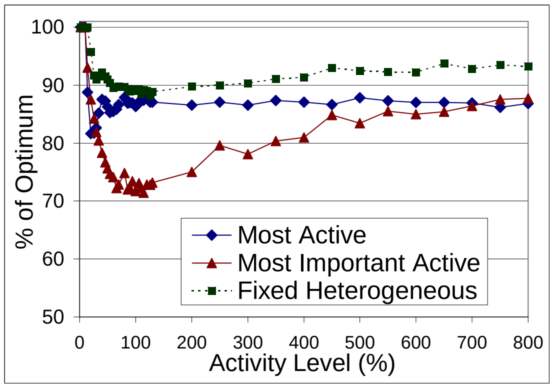


(a) Conversation Value Ratio 1:3 (b) Conversation Value Ratio 1:5

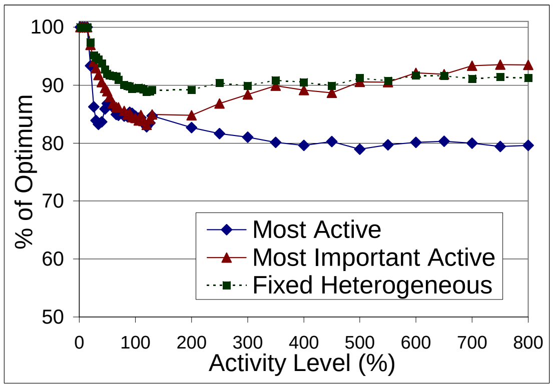
(c) Conversation Value Ratio 1:8

Figure 9.7: Fixed Heterogeneous Policies - Value 75% & Volume 25%

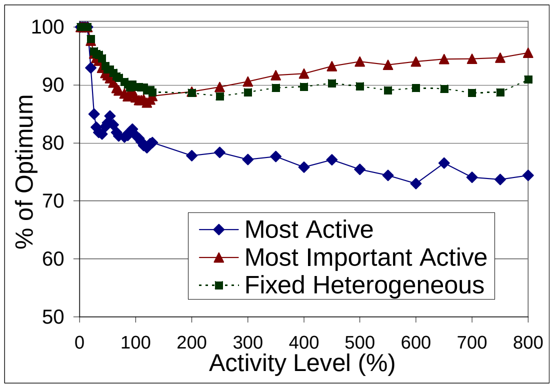
We expected heterogeneous overhearing to act as an intermediate between the two extremes of value and volume: In settings where *MostImportantActive* policy



(a) Conversation Value Ratio 1:3



(b) Conversation Value Ratio 1:5

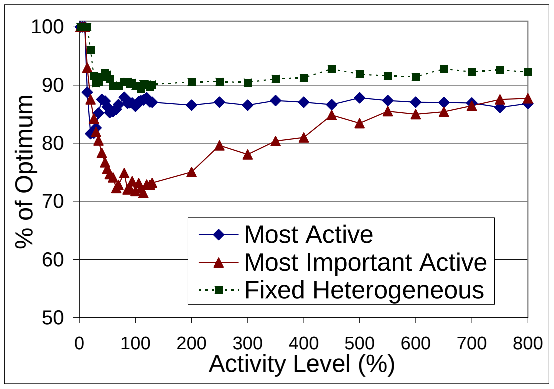


(c) Conversation Value Ratio 1:8

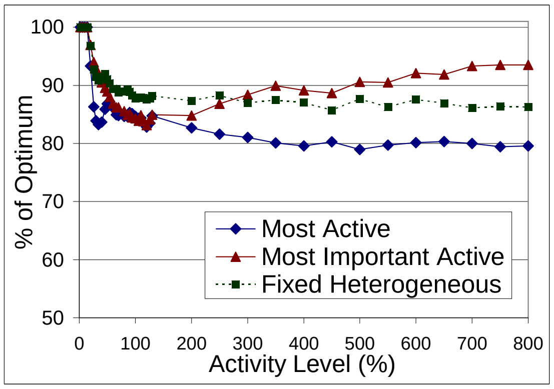
Figure 9.8: Fixed Heterogeneous Policies - Value 50% & Volume 50%

outperforms the *MostActive* policy, heterogeneous policies were expected to outperform the *MostActive* policy, while still performing poorer than the *MostImportantActive* policy. Similarly, in settings where *MostActive* policy performs better than the *MostImportantActive* policy, a vise versa behavior was expected. Furthermore, heterogeneous policies were expected to perform closer to the *MostActive* policy in combinations with higher percentage of agents following the *MostActive* policy, and to show closer performance to the *MostImportantActive* policy in groups where most agents use the *MostImportantActive* policy.

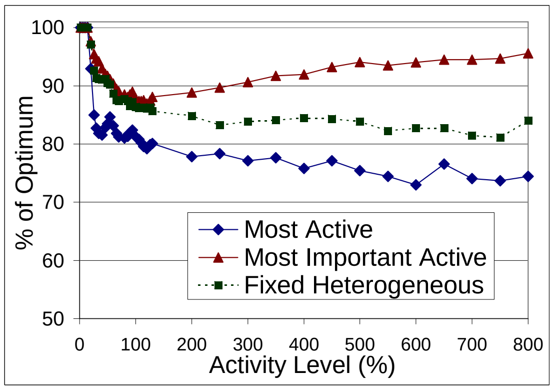
Nonetheless, Figures 9.7, 9.8 and 9.9 show a surprising behavior of fixed heterogeneous policies. It can be seen that heterogeneous policies mostly outperform both the *MostImportantActive* and the *MostActive* policies. Thus, using heterogeneous policies, we are witnessing a condition where *the sum of the parts is bigger than the whole*.



(a) Conversation Value Ratio 1:3



(b) Conversation Value Ratio 1:5



(c) Conversation Value Ratio 1:8

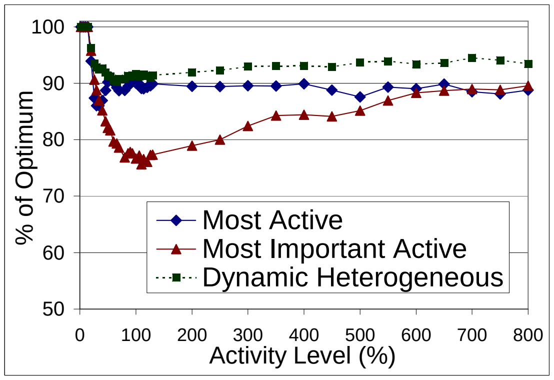
Figure 9.9: Fixed Heterogeneous Policies - Value 25% & Volume 75%

9.3.2 Dynamic Heterogeneous Policies

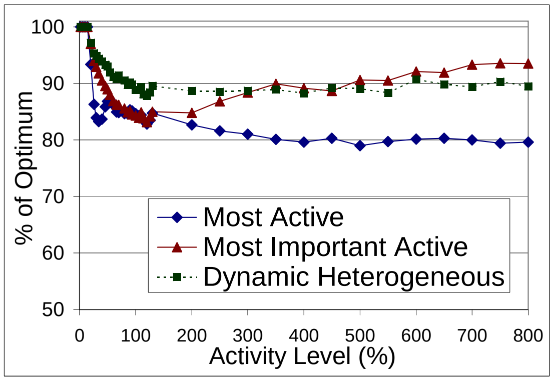
Given the improvement in performance achieved by combining the *MostImportantActive* and the *MostActive* policies in fixed heterogeneous overhearing, we sought to apply this combination in a more flexible fashion. Fixed heterogeneous policies may perform well in some settings of conversation systems, while performing poorly in others. However, being bounded to a static division of an overhearing group, these policies have zero-ability to adjust.

Therefore, instead of statically defining for each agent in advance which overhearing policy it is about to follow, we make this decision dynamically for each overhearing agent. Deciding on the policy the agent is about to follow, we choose between two options. We decide whether the corresponding agent will overhear the most important active agent choosing to follow the *MostImportantActive* policy, or will it target the most active agent according to the *MostActive* policy. Moreover, we decide whether it is better to overhear x conversations by the most important active agent $agent_i$ or y conversations by the most active agent $agent_j$. Usually, y will be greater than x, but the value of conversations by $agent_i$ will be higher than the ones by $agent_j$. Thus, the dilemma.

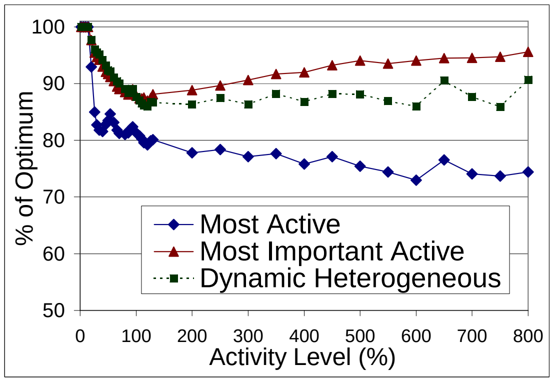
This decision is made based on communicating agents' past performance–the average value of conversations in which the agents were overheard earlier (as recognized by the conversation recognition process, see Chapter 8, Section 8.1.2). Given that this section addresses centralized selective overhearing, we assume that this memory is *shared* to all agents in the overhearing group. The distributed case of *individual* memory will be addressed further in Chapter 10.



(a) Conversation Value Ratio 1:3



(b) Conversation Value Ratio 1:5

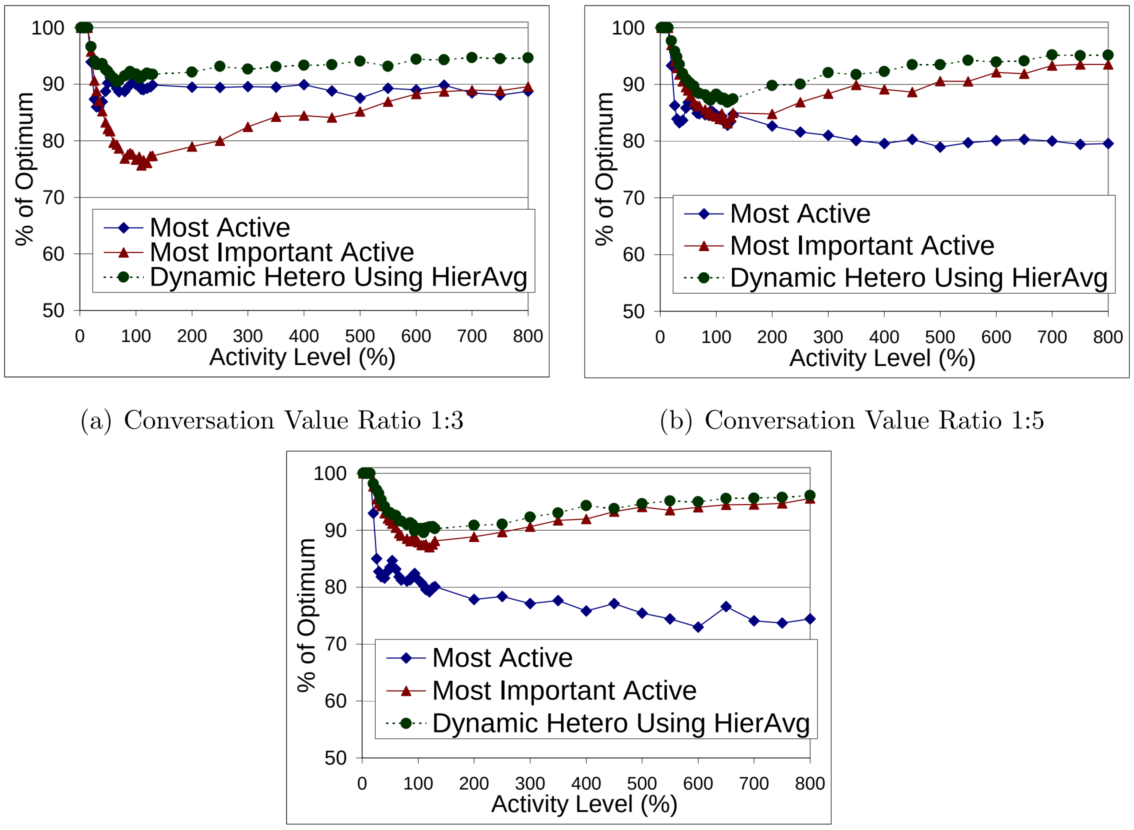


(c) Conversation Value Ratio 1:8

Figure 9.10: Dynamic Heterogeneous Policies

Figures 9.10-a,b,c show the performance of the *DynamicHeterogeneous* policy in comparison to the *MostImportantActive* and the *MostActive* policies under different configurations of conversation value ratio. Although we expected the *DynamicHeterogeneous* policy to always outperform both the *MostImportantActive* and the *MostActive* policies, it can be seen that this clearly isn't the case. Indeed, Figure 9.10-a (conversation value ratio of 1:3) shows that the *DynamicHeterogeneous* policy outperforms both the *MostImportantActive* and the *MostActive* policies at all activity levels. However, in Figure 9.10-b (value ratio of 1:5), the *DynamicHeterogeneous* policy performs poorer than the *MostImportantActive* policy in high activity levels. Furthermore, in Figure 9.10-c (value ratio of 1:8), the *MostImportantActive* policy outperforms the *DynamicHeterogeneous* policy in all activity levels.

However, we argue that this behavior is caused only due to the incompleteness of information on which the value-volume decision is based. In Figures 9.10-a,b,c this decision is based on the shared memory of communicating agents' past performance. At start, this memory contains no knowledge. Only with time, by overhearing conversations, overhearing agents acquire the knowledge on average value of agents' conversations. Thus, some decisions made based on this memory may be inaccurate. Figures 9.10-a,b,c show that the "price" of a mistake in value-volume decision is higher in higher activity levels and more significant value ratios.



(a) Conversation Value Ratio 1:3 (b) Conversation Value Ratio 1:5

(c) Conversation Value Ratio 1:8

Figure 9.11: Dynamic Heterogeneous Policies using Hierarchy Average Conversation Values

To prove this claim, we performed the same set of experiments, only now using more accurate information to make the value-volume decision. Here, this decision was made based on the average value of conversations at different hierarchy levels. Figures 9.11-a,b,c show that the *DynamicHeteroUsingHierAvg* policy (same as the *DynamicHeterogeneous* policy but using average conversation values of hierarchy levels) outperforms both the *MostImportantActive* and the *MostActive* policies in all activity levels for all conversation value ratios.

Nevertheless, this information is usually unaccessible to overhearing agents. Thus, we consider the *DynamicHeterogeneous* policy (shown in Figures 9.10-a,b,c) to be the most effective centralized selective overhearing policy. Furthermore, we use it as a baseline for comparison with distributed overhearing policies presented in Chapter 10.

9.4 Discussion

In this chapter, we presented an empirical study of centralized selective overhearing policies for hierarchically-structured organizations. Using the model for selective overhearing in such organizations and its simulation presented in Chapter 8, we performed an extensive set of experiments examining several centralized overhearing policies appropriate for hierarchical settings.

Based on these experiments, we make several important qualitative conclusions:

- **Active vs. Static Policies**. Knowledge of which agents are active, facilitating active policies, is crucial in organizations with low conversation activity. However, as activity level rises, the advantage of active policies disappears.

- **Value vs. Volume Policies**. Comparing the value and the volume policies, a classical tradeoff have been found. However, the performance of these two policies with respect to each other was found to be surprisingly robust to changes in the shape of the organization (pyramidal, diamond-shaped, etc.), the number of agents overheard, and the number of hierarchical levels in the organization. Instead, their relative performance changes qualitatively only with changes in how much the conversations at top levels are more important than the conversations in low levels of the organization.

- **Heterogeneous Policies.** Although the value-volume tradeoff has been found to be robust to many characteristics of the monitored organizations, it is still influenced by some of these characteristics (mainly by the conversation value ratio, but also by the activity level to a smaller degree). Unfortunately, these characteristics are not always known in advance. Thus, committing to one of these policies "blindly" causes either a very good or a very poor performance. To mediate the gap between the two extremes, we studied a combination of both policies where some overhearing agents follow the value policy, while others follow the volume overhearing policy. To our surprise, we found that combining the two policies has the potential to outperform each of the original policies separately for any monitored organization, and thus to be a potentially effective centralized overhearing policy independent of any influencing characteristics of the monitored organization.

This chapter addressed centralized selective overhearing policies appropriate for hierarchical organizations. In the next chapter (Chapter 10), we examine the behavior of distributed selective overhearing policies in these settings.

Chapter 10

Empirical Study of Distributed Policies

Up to this point, we conducted an empirical study of selective overhearing in hierarchical organizations focusing on centralized policies (Chapter 9). However, applying centralized policies in large-scale distributed settings can be problematic. Tendency towards distribution exists both in the real-world settings and in the multi-agent systems. Growth in the number of global international companies on one hand, and of open distributed multi-agent applications on the other, constitute the development in this direction in both fields. In such settings, a centrally-located overhearer (which is equivalent to an overhearing group coordinated using centralized policy) might not be able to monitor communications performed by geographically-distributed agents. Instead, a team of cooperating overhearing agents, acting in a distributed manner, should be used. This chapter is dedicated to the first steps towards studying distributed selective overhearing of hierarchical organizations.

10.1 Centralized vs. Distributed Policies

In Chapter 9, we have focused on centralized selective overhearing. We have shown heterogeneous overhearing policies to be effective centralized policies. However, all these policies rely on a number of assumptions (some of which were implicit up to this point) possible only in centralized settings. In this chapter, we are going to tackle these assumptions moving from centralized to distributed selective overhearing.

As already mentioned, we are going to address the *DynamicHeterogeneous* policy as a baseline centralized policy. This policy relies on the following assumptions: (i)

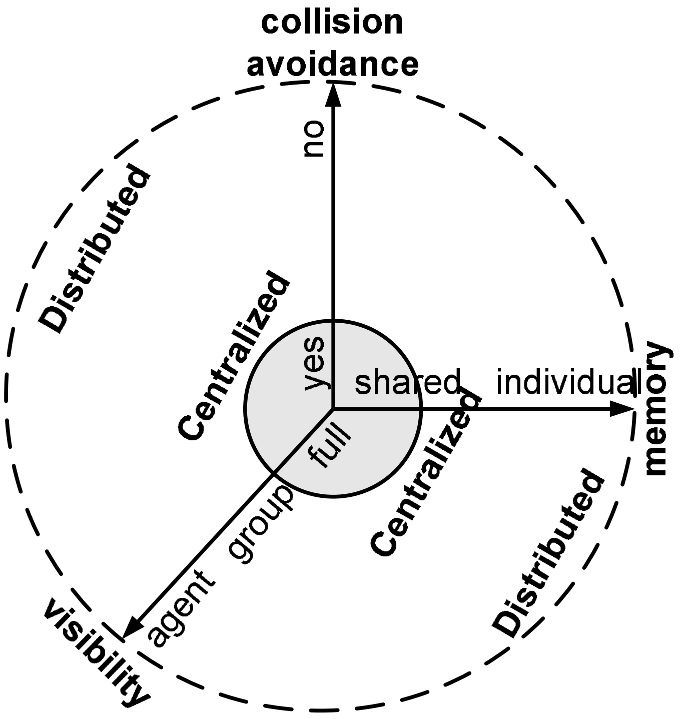


Figure 10.1: Centralized vs. Distributed Policies

full visibility–all overhearing agents are aware of all conversations carried out in the monitored organization; (ii) *shared memory*–the value-volume decision is made based on memory shared by all overhearing agents; and (iii) *collision avoidance*–a target chosen by one overhearing agent, can not be chosen by another overhearer.

Thus, moving from centralized policies to distributed ones, we simply reduce overhearing agents' collaboration along these three dimensions. Figure 10.1 summarizes this transition. Addressing the memory dimension, we distinguish between two types of memory. The first is *shared* by all overhearing agents, whereas the second type of memory is *individual* for each overhearer.

As to the visibility dimension, we distinguish between three types of visibility. The first is the *full visibility* already explained above. The second, i.e. *group visibility*, assumes that overhearing agents are only aware of the conversations that are carried out by their targets (they are not aware of the conversations carried out by agents that are not being overheard). The last visibility type is *agent visibility*. Here, as opposed to group visibility, overhearing agent is only aware of conversations committed by its target, but not by targets overheard by other overhearers. However, the overhearing agent has a limited visibility of conversations carried out by other communicating agents: It is aware of conversations that these agents hold with its own target.

Finally, in the collision avoidance dimension, we distinguish between two extremes. On one hand, the case where all collisions are avoided, and, on the other hand, the case where collisions are allowed and there is no collision avoidance mechanism applied.

Figure 10.1 summarizes the differences between centralized and distributed policies. Centralized policies assume full visibility, shared memory and collision avoidance. On the other hand, distributed policies assume the other extreme: agent visibility, individual memory and no collision avoidance. Any other combination is considered to be partially centralized and partially distributed. In the remaining part of this chapter, we show a transition from centralized overhearing policies to the distributed ones along the three dimensions described above.

10.2 Memory Dimension

Exploring selective overhearing policies with respect to memory dimension, we first assume full visibility and collision avoidance. However, the later experiments, presented in Sections 10.3-10.4, annul this assumption as well.

Following this initial assumption, we introduce two selective overhearing policies–*FullVis-ShrdMem-CollAvd* and *FullVis-IndMem-CollAvd*. The first is in fact the *DynamicHeterogeneous* policy presented in Chapter 9, Section 9.3.2. It uses *shared* memory of communicating agents' past performance as a basis for the value-volume decision. To remind the reader, the value-volume decision considers whether the agent will target most-important agents following the *MostImportantActive* (Value) policy or whether it will target most communicative agents using the *MostActive* (Volume) policy (see Chapter 9, Section 9.3.2 for further details).

In contrast, the *FullVis-IndMem-CollAvd* policy uses *individual* memory. According to this policy, each overhearing agent has only the memory of conversations it overheard in the past, without any knowledge on conversations overheard by other overhearers. Thus, the basis for the value-volume decision for each agent lies in its own past experience.

The distinction between the two types of memory is important mainly due to the requirements of their maintenance. Maintaining a consistent and accurate shared memory requires broadcasting information value of each overheard conversation to all overhearing agents, whereas individual memory requires no communications between overhearing agents.

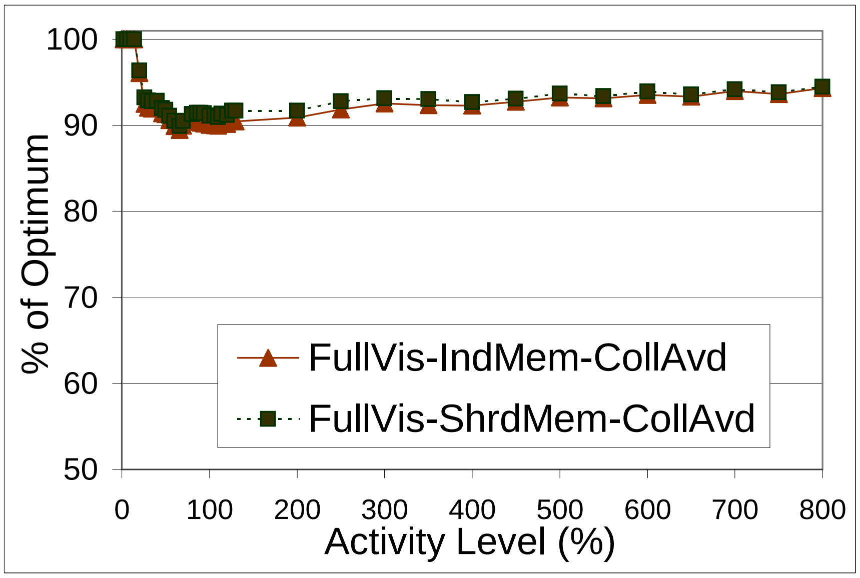


Figure 10.2: Full Visibility - Memory Effect

Figure 10.2 shows the performance of the *FullVis-ShrdMem-CollAvd* and the *FullVis-IndMem-CollAvd* policies. It can be seen that an overhearing group performs better using shared memory than the case where an individual memory is applied. However, the gap between the performance of the two policies appears to be insignificant.

10.3 Visibility Dimension

The experiments presented so far all assumed *full* visibility, i.e. all overhearing agents are aware of all the conversations carried out in the monitored organization at any given moment. In this section, we are going to confront this assumption by first assuming the transition from *full* to *group* visibility in Section 10.3.1, and then addressing the transition to the even more restricted case of *agent* visibility in Section 10.3.2.

10.3.1 Full vs. Group Visibility

Full visibility is usually unobtainable. Thus, we now assume *group* visibility. According to this visibility, overhearing agents are only aware of the conversations carried out by agents that are being overheard. Still, each overhearing agent is aware of both the conversations carried out by its target and of the conversations by targets overheard by other overhearers. Group visibility restricts the visibility of an organization to the limits of an overhearing group. However, it still assumes that overhearing agents collaborate informing each other on the overheard conversations.

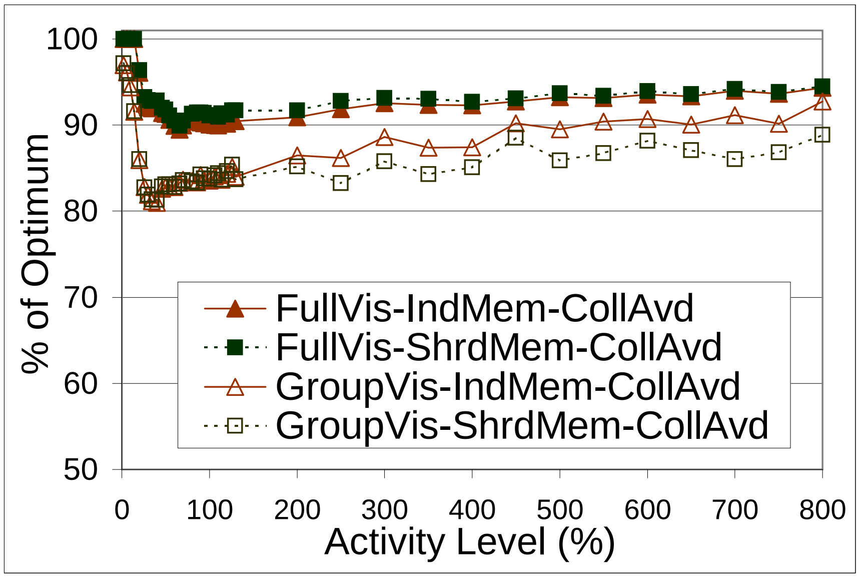
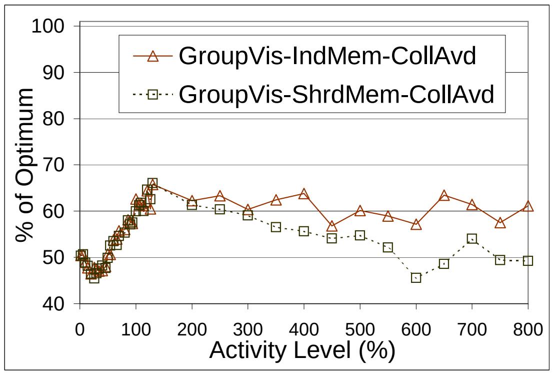


Figure 10.3: Group Visibility - Memory Effect

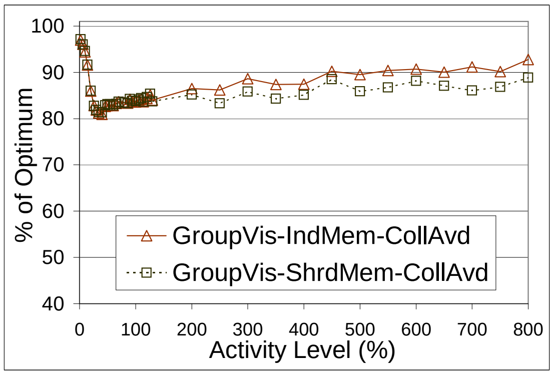
Figure 10.3 shows the performance of the two policies, discussed in Section 10.2, with respect to group visibility. These policies are called *GroupVis-ShrdMem-CollAvd* and *GroupVis-IndMem-CollAvd* respectively. Comparing these policies to the ones relying on full visibility, we can see a degradation in performance. This conclusion is straightforward to some extent since the transition from full to group visibility reduces the knowledge on the monitored organization.

However, a more surprising result comes from contrasting the *GroupVis-ShrdMem-CollAvd* and the *GroupVis-IndMem-CollAvd* policies. In contrast to the performance of the two policies with respect to full visibility, it is the policy using individual memory that outperforms the policy using shared memory under condition of group-visibility settings.

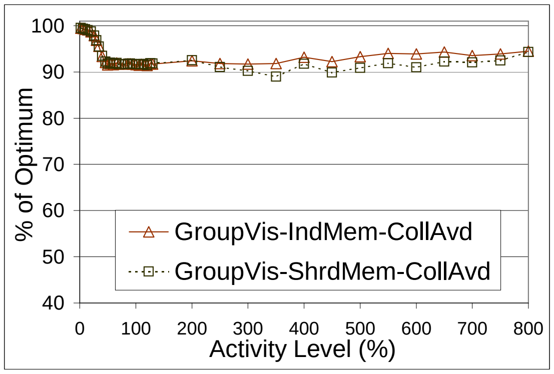
Moreover, group visibility assumes that the overhearing agent is both aware of the conversations carried out by its target and of the conversations carried out by targets overheard by other overhearers. Thus, group visibility depends on the number of overhearing agents. In the following set of experiments, we tested the surprising behavior of the *GroupVis-ShrdMem-CollAvd* and the *GroupVis-IndMem-CollAvd* policies with respect to different number of overhearers–measured as coverage %, i.e. the ratio of the number of overhearing agents to the number of potential targets.



(a) Coverage: 5%



(b) Coverage: 30%



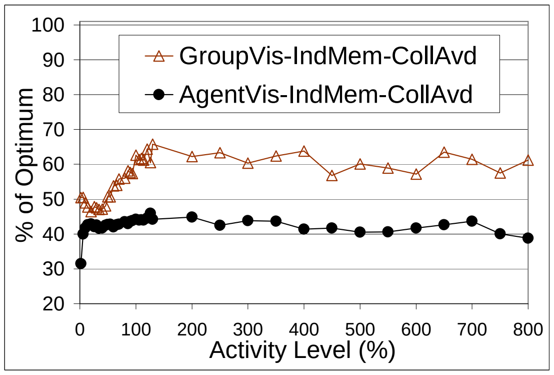
(c) Coverage: 50%

Figure 10.4: Group Visibility - Shared vs. Individual Memory - Coverage Effect

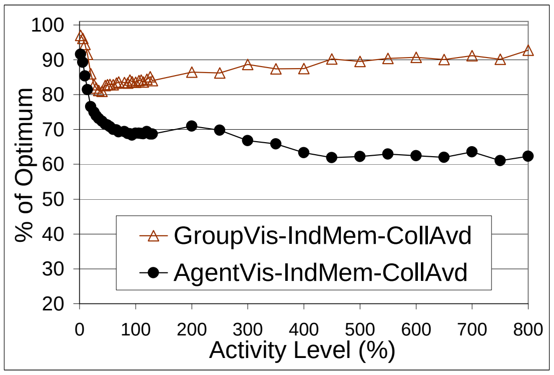
Figure 10.4-b shows the performance of the two group-visibility policies with respect to the default overhearing coverage of 30%, while Figures 10.4-a and c show the performance results under condition of two additional coverage ratios (5% and 50% respectively). Figures 10.4-a,b and c all show that the surprising behavior of individual memory outperforming the shared memory, found in group-visibility settings, is consistent in all levels of overhearing coverage.

10.3.2 Group vs. Agent Visibility

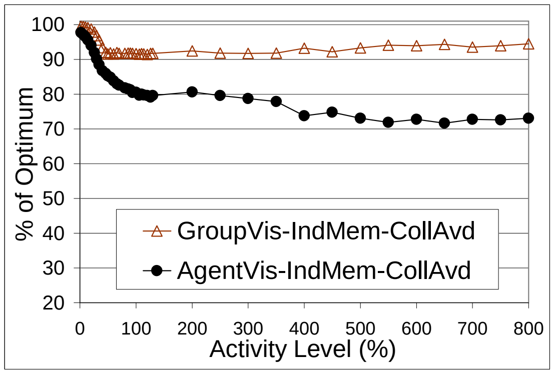
Moving further towards distributed selective overhearing, we now assume *agent* visibility. As opposed to group visibility, overhearing agent is only aware of conversations committed by its target, but not by the targets overheard by other overhearers. However, the overhearing agent has a limited knowledge on conversations held by other communicating agents: It is aware of the conversations that these agents hold with its own target.



(a) Coverage: 5%



(b) Coverage: 30%



(c) Coverage: 50%

Figure 10.5: Group vs. Agent Visibility - Individual Memory - Coverage Effect

Figures 10.5-a,b and c compare selective overhearing policies in group- and agent-visibilities. However, considering the surprising result shown earlier in Figures 10.4-a,b and c (where individual memory outperforms shared memory), we only address policies using individual memory, ignoring shared memory policies in these settings. This step also supports our desired goal of moving towards distributed selective overhearing policies where the overhearing agents are loosely-coupled.

Again, we can see that the transition from group visibility to agent visibility results in poorer performance. In all coverage ratios (Figures 10.5-a,b and c), the *GroupVis-IndMem-CollAvd* policy outperforms the *AgentVis-IndMem-CollAvd* policy. This result is similar to the effect caused by the transition from full visibility to group visibility. In both cases, the available information on the monitored organization becomes smaller with each transition, and thus causes the degradation in performance.

10.4 Collision Avoidance Dimension

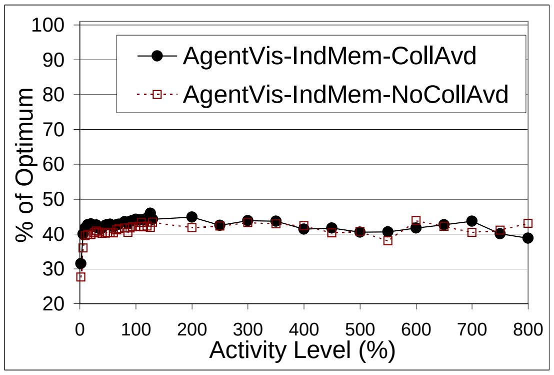
The final subject of our research in this chapter of the thesis is the influence of collision avoidance on the performance of selective overhearing policies. Collision is defined as a state where two or more overhearing agents target the same communicating agent at the same time. In centralized settings, such collisions can easily be avoided since all agents are coordinated by a single centralistic authority. In contrast, overhearing agents operating in distributed settings must handle collisions on their own. Therefore, collision avoidance might be a time-consuming activity for an overhearing agent.

In Section 10.4.1, we explore the general effect of collision avoidance comparing the case where all collisions are avoided with the case where no collision mechanism is applied at all, whereas Section 10.4.2 studies the case of partial collision avoidance, i.e. where only some of the collisions are avoided. In both sections, we assume agent visibility and the use of individual memory.

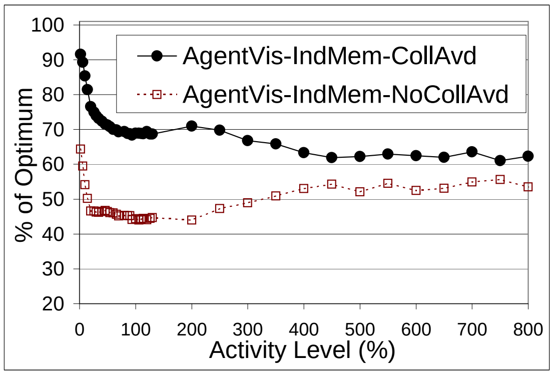
10.4.1 None vs. Full Collision Avoidance

Figures 10.6-a,b and c show the performance of selective overhearing policies with respect to the two extremes: one where collisions are always avoided, i.e. the *AgentVis-IndMem-CollAvd* policy, and other where collisions are allowed–the *AgentVis-IndMem-NoCollAvd* policy. Moreover, in the *AgentVis-IndMem-NoCollAvoid* policy, even if a collision is detected, i.e. overhearing agent chooses a target already overheard by another overhearer, the overhearer does not change its selected target. On the other hand, in the *AgentVis-IndMem-CollAvoid* policy, in case of collision, the overhearing agent chooses a second best target, then the third-best target and so on in case of additional collisions. Finally, if it has no more desired targets, it simply chooses a random one.

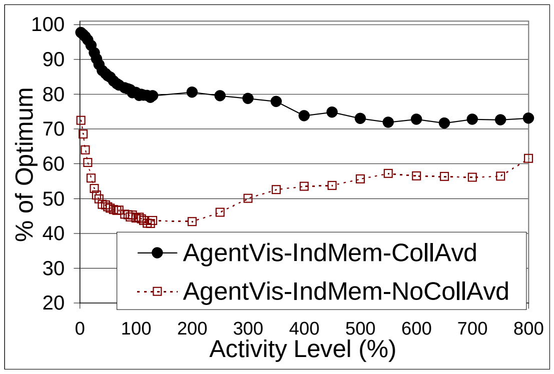
Since collisions depend on the number of overhearers, Figures 10.6-a,b and c show the performance of these two overhearing policies with respect to different overhearing coverage percentage (5%, 30% and 50% respectively). Indeed, it can be seen that collision avoidance has a significant effect on the performance of overhearing policies. Furthermore, this effect becomes more significant in larger overhearing groups. Although in settings with low overhearing coverage no difference is witnessed (with or without collision avoidance), it becomes highly significant to avoid collisions as the number of overhearing agents increases due to the higher probability of collisions in such settings.



(a) Coverage: 5%



(b) Coverage: 30%



(c) Coverage: 50%

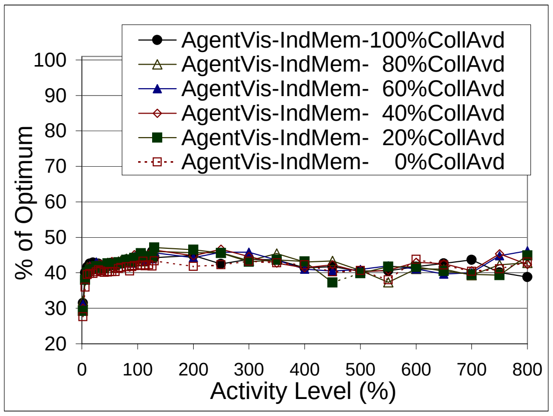
Figure 10.6: Agent Visibility - Collision Avoidance Effect

10.4.2 Partial Collision Avoidance

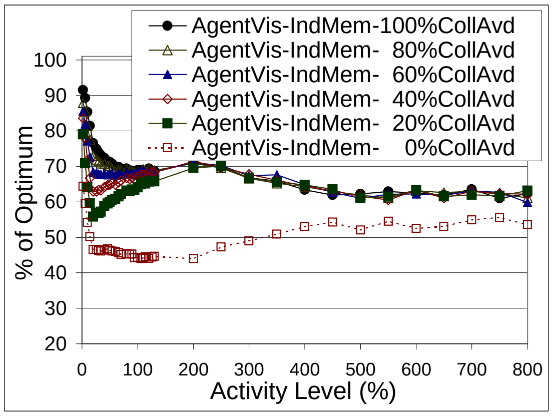
As a final step of our research, we examined different levels of collision avoidance. Here, we compare overhearing policies where overhearing agent seeks to avoid only a certain amount of occurring collisions.

Figures 10.7-a,b and c show the corresponding results. The *AgentVis-IndMem-p%CollAvoid* policy represents a selective overhearing policy according to which the overhearing agent chooses to avoid only *p* percent of occurring collisions. This *p* percent ranges from 0% to 100% (with a 20% hop). The *AgentVis-IndMem-0%CollAvoid* and the *AgentVis-IndMem-100%CollAvoid* policies correspond to the *AgentVis-IndMem-NoCollAvoid* and the *AgentVis-IndMem-CollAvoid* policies above.

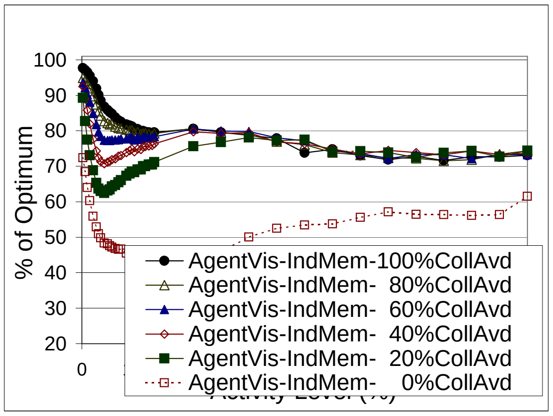
Again, it can be seen that in settings with low overhearing coverage collision avoidance has no significant effect (Figures 10.7-a). Similarly, different levels of collision avoidance loose their impact as activity levels rise (Figures 10.7-b and c). However, in low and medium activity levels, collision avoidance and its percentage are important. Still, it can be seen that the performance boost, achieved due to the higher levels of collision avoidance, is not always significant.



(a) Coverage: 5%



(b) Coverage: 30%



(c) Coverage: 50%

Figure 10.7: Agent Visibility - Collision Avoidance % Effect

10.5 Discussion

In Chapter 9, we discussed in details centralized overhearing policies of hierarchical organizations under the restriction of selectivity. On the other hand, in this chapter, we explored the transition from centralized selective overhearing policies to distributed ones.

In effective centralized policies, we identified three dimension in which overhearing agents are centrally-coordinated. These dimensions are memory, visibility and collisions avoidance. The potential targets are determined based on the visibility of their current conversations and the memory of their prior conversations, while collisions avoidance mechanism coordinates overhearing agents to prevent two different overhearers to overhear the same target. An effective centralized policy assumes full visibility, shared memory and no collisions.

Transitioning to distributed policies, we incrementally increased the autonomy degree of overhearing agents (within the overhearing group) along these three dimensions. Addressing memory dimension, we considered the transition from *shared* to *individual* memory. As for visibility, we distinguish between *full* visibility where each overhearing agent is aware of all conversations, *group* visibility where each overhearer is aware only of the conversations overheard by the overhearing group and *agent* visibility–being aware only of the conversations it overhears. Finally, we refer to cases where all, none and some collisions are avoided. Clearly, the fully distributed selective overhearing policy assumes individual memory, agent visibility and no collision avoidance.

Our experiments studied the changes in performance of overhearing policies caused by the changes in coordination of the overhearing groups along these three dimensions. We come to conclusion that some of the changes have greater effect than others. For instance, the transition from shared to individual memory does not influence the performance of selective overhearing. This conclusion is important since maintaining shared memory causes frequent communications between overhearing agents, and thus might burden the communication network. In addition, we show that it is sufficient to solve some of collision and not all of them. Again, this conclusion is important since collision avoidance is a time-consuming activity that might be problematic in real-time settings.

Still, the transition from centralized to distributed overhearing policies causes a significant decrease in performance (mainly due to decrease in visibility). However, we believe that studying the factors influencing this transition, as done in this thesis, will help us to bridge this gap in the future. Furthermore, identifying the effect of each factor will enable to focus the effort on the most influencing factors, while not wasting valuable resources in vague.

Chapter 11

Final Remarks

Recent multi-agent systems are often built applying an open, distributed design. These systems involve various challenges of monitoring geographically-distributed and independently-built multiple agents. *Monitoring by overhearing* [Kaminka *et al.*, 2002] is a monitoring approach particularly suited for open distributed MAS settings. Here, an overhearing agent monitors the exchanged communications between the system's agents. These overheard routine communications are used to independently assemble and infer the needed monitoring information.

Previous investigations of overhearing have demonstrated a range of overhearing techniques and applications. Overhearing was used for maintaining organizational and situational awareness [Novick and Ward, 1993, Legras, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005], debugging and detecting inconsistencies [Poutakidis *et al.*, 2002, Rossi and Busetta, 2004, Rossi and Busetta, 2005], monitoring progress [Kaminka *et al.*, 2002] and discovering opportunities for providing advice [Aiello *et al.*, 2001, Busetta *et al.*, 2001]. Given these and other potential applications, we believe that the principle of overhearing is of great importance both in the fields of commercial and military implementations along with broad research perspectives.

However, previous works mainly focus on the potential applications of overhearing. In doing so, these investigations often rely on problematic assumptions related to the fundamentals of overhearing. Our work systematically tackles these problematic assumptions addressing the fundamental building blocks of overhearing.

A theme that runs through our research is the focus on the challenges of using overhearing in large-scale multi-agent settings. We first addressed the building block of conversation representation providing a scalable representation of multi-agent conversations for overhearing. Then, building on it, we explored the building block of conversation recognition addressing the cases of overhearing where the overhearer can not overhear all exchanged messages. Finally, addressing the cases where not just messages, but entire conversations are not accessible to the overhearing agent, we empirically studied the building block of selective overhearing. In the remaining part of this chapter, we summarize our work related to each of these building blocks, draw the main conclusions and provide possible directions for future research.

11.1 Representing Conversations for Overhearing

Petri nets have recently been shown to provide a viable representation approach for modelling multi-agent conversations [Cost *et al.*, 1999, Cost *et al.*, 2000, Nowostawski *et al.*, 2001, Mazouzi *et al.*, 2002]. However, radically different approaches have been proposed to using Petri nets for modelling multi-agent conversations. Yet, the relative strengths and weaknesses of the proposed techniques have not been examined.

Our work in Part I of the thesis introduces a novel classification of previous investigations and then compares these investigations addressing their scalability and appropriateness for monitoring via overhearing. Based on the insights gained from the analysis, we have developed a novel Petri net representation of multi-agent conversations. This representation technique offers significant improvements (compared to previous approaches) in terms of scalability, and thus is more appropriate for representing multi-agent conversations in large-scale settings. We also show this representation to be particularly suitable for monitoring via overhearing.

We systematically show how this representation covers essentially all the features required to model complex multi-agent conversations, as defined by the FIPA conversation standards [FIPA Specifications, 2005c]. These include simple & complex interaction building blocks, communicative act attributes and multiple concurrent conversations, nested & interleaved interactions and temporal aspects of conversations. Previous approaches could handle some of these examples (though with reduced scalability), but none was shown to cover all the required features.

Finally, we developed a skeleton procedure for semi-automatically converting an AUML protocol diagrams (the chosen FIPA representation standard) to an equivalent Petri net representation. We have demonstrated its use on a challenging FIPA conversation protocol, which was difficult to represent using previous approaches.

Naturally, some issues remain open for future work. We believe that the automatic conversion of human-readable AUML protocol diagrams to their machine-readable Petri net representations is extremely important in the view of the existing automatic techniques for debugging [Poutakidis *et al.*, 2002], validation and verification [Desel *et al.*, 1997], deadlock detection [Khomenco and Koutny, 2000], etc. Though our work addressed such automatic conversion for AUML protocol diagrams representing two agent roles, the general *n-agent* version still remains a challenging problem open for future research.

11.2 Conversation Recognition

Our work on conversation recognition was the first to address this key step in overhearing. Most previous investigations on overhearing either simply ignore it or assume it can be done without providing any implementation details.

In contrast, our work in Part II of the dissertation formally addresses conversation recognition. Since most previous works address overhearing in context of specific applications, our first goal was to provide a formal model of overhearing unrelated to any specific task. Based on the proposed model, we have formalized the problem of conversation recognition and provided algorithms for handling it both in lossless and lossy settings. Furthermore, we have analyzed the efficiency of the introduced algorithms with respect to their applicability in large-scale MAS.

Still, we believe that our work constitutes only the first steps in this direction. We have analyzed conversation recognition for the most common case where the overhearing agent does not overhear some of the exchanged messages due to noise, its relative position or some other factor. In our opinion, it is also important to consider additional erroneous cases where the sequence of overheard messages is different from the sequence of messages exchanged in the actual conversation. These possible differences are summarized in Table 6.1 in Chapter 6, Section 6.3.

In context of conversation recognition, we also distinguish two additional directions open for future research. The first is analyzing the appropriateness of different types of algorithms for conversation recognition. The algorithms proposed in our work are *forward chaining* algorithms, i.e. algorithms that examine the sequence of overheard messages from first to last. In contrast, it is also possible to examine the same sequence of overheard messages in a reverse order–from last to first. This sort of algorithms called *back chaining* algorithms. The works by

[Klein, 1997, Fan and Yen, 2005] showed that certain tradeoffs exist using the two types of algorithms. We believe that considering the application of the two types of algorithms for conversation recognition and analyzing their tradeoffs in this context offer an interesting research ground.

Another direction open for future research is conversation recognition of multi-party communications recently gaining attention in multi-agent community [Kumar *et al.*, 2000, Traum and Rickel, 2002, Dignum and Vreeswijk, 2003]. As opposed to the point-to-point communications between two agents, multi-party communications involve a group of communicating agents where each agent can send a message at any given time, each agent is affected differently by the exchanged communications, etc. Our algorithms, presented in Chapter 7, handle conversation recognition in context of pairwise communications. However, the specific challenges related to conversation recognition of groupwise communications remain unattained. Identifying this opportunity, [Fan and Yen, 2005] made the first steps in this direction. Nonetheless, many challenges still remain open.

11.3 Selective Overhearing

Selective overhearing addresses the specific characteristics of overhearing in large-scale MAS. In such settings, it is reasonable to assume that the available overhearing resources are bound to be limited. Thus, the assumption that *all* relevant inter-agent communications can be overheard (made by all previous investigations) can be challenged. Instead, the overhearing agent must carefully choose its targets, since only a subset of those can be overheard.

We propose to use organizational knowledge of the monitored settings to make this decision. In our work, the overhearing agent decides on choosing one target over the other based on factors such as its organizational role within the monitored system and the specific characteristics of its conversations compelled by this role. We believe that in the future it would be interesting to investigate alternative criterions for making this decision.

Our work studies overhearing of hierarchical organizations. Thus, we first propose a model of selective overhearing taking into account both the limitations of selectivity and the specific characteristics of such organizations. We focus on hierarchical organizations since those are widely common both in real-world (e.g., many corporates) and in multi-agent applications (e.g., [So and Durfee, 1996, Yadgar *et al.*, 2003]).

Still, it would be rather interesting to study overhearing of different types of organizations used in multi-agent community such as holarchies, coalitions, teams, congregations, societies, federations and matrix-organizations [Horling and Lesser, 2004]. The choice of organizational paradigm depends on the application. While suitable for some tasks, the same organizational paradigm might be less appropriate for implementing some other tasks [Corkill and Lander, 1998, Horling and Lesser, 2004]. Accordingly, in overhearing different types of organizations, we would have to consider these differences between the various organizational paradigms.

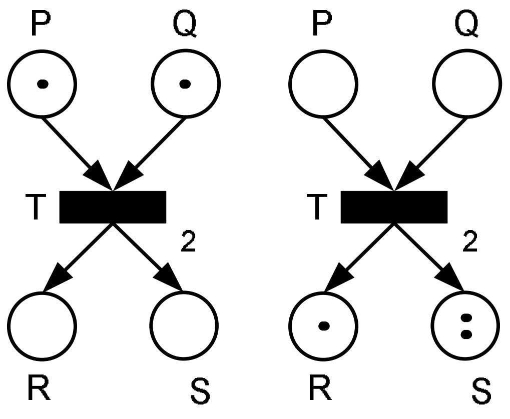
Based on the proposed model for hierarchical organizations, we empirically study various centralized and distributed selective overhearing policies. In doing so, we examine overhearing performed by a group of cooperative agents tackling the assumption by previous investigations that overhearing is either done by a single centrally-located overhearer or by a group of non-collaborating overhearing agents that perform overhearing out of their own interest. Given the interest in teamwork [Pynadath and Tambe, 2003, Scerri *et al.*, 2004, Paruchuri *et al.*, 2004], we believe that our work in this direction only constitutes the first steps in examining the implications of applying the concept of teamwork in overhearing performed by a group of collaborative overhearers.

Appendix A

A Brief Introduction to Petri Nets

Petri nets [Petri Nets site, 2005] are a widespread, established methodology for representing and reasoning about distributed systems, combining a graphical representation with a comprehensive mathematical theory. One version of Petri nets is called Place/Transition nets (PT-nets) [Reisig, 1985]. A PT-net is a bipartite directed graph where each node is either a place or a transition (Figure A.1). The net places and transitions are indicated through circles and rectangles respectively. The PT-net arcs support only place $\rightarrow$ transition and transition $\rightarrow$ place connections, but never connections between two places or between two transitions. The arc direction determines the input/output characteristics of the place and the transition connected. Thus, given an arc, $P \rightarrow T$, connecting place P and transition T, we will say that place P is an input place of transition T and vice versa transition T is an output transition of place P. The $P \rightarrow T$ arc is considered to be an output arc of place P and an input arc of transition T.

A PT-net place may be *marked* by small black dots called *tokens*. The arc expression is an integer, which determines the number of tokens associated with the corresponding arc. By convention, an arc expression equal to 1 is omitted. A specific transition is *enabled* if and only if its input places *marking* satisfies the appropriate arc expressions. For example, consider arc $P \rightarrow T$ to be the only arc to connect place P and transition T. Thus, given that this arc has an arc expression 2, we will say that transition T is enabled if and only if place P is marked with two tokens. In case the transition is enabled, it may *fire/occur*. The transition occurrence removes tokens from the transition input places and puts tokens to the transition output places as specified by the arc expressions of the corresponding input/output arcs. Thus, in Figures A.1-a and A.1-b, we demonstrate PT-net marking before and after transition firing respectively.



(a) Before firing (b) After firing

Figure A.1: A PT-net example.

Although computationally equivalent, a different version of Petri nets, called *Colored Petri nets* (CP-nets) [Jensen, 1997a, Jensen, 1997b, Jensen, 1997c], offers greater flexibility in compactly representing complex systems. Similarly to the PT-net model, CP-nets consist of net places, net transitions and arcs connecting them. However, in CP-nets, tokens are not just single bits, but can be complex, structured, information carriers. The type of additional information carried by the token, is called *token color*, and it may be simple (e.g., an integer or a string), or complex (e.g. a record or a tuple). Each place is declared by a *place color* set to only match tokens of particular colors. A CP-net place marking is a token *multi-set* (i.e., a set in which a member may appear more than once) corresponding to the appropriate place color set. CP-net arcs pass token multi-sets between the places and transitions. CP-net arc expressions can evaluate token multi-sets and may involve complex calculation procedures over token *variables* declared to be associated with the corresponding arcs.

The CP-net model introduces additional extensions to PT-nets. *Transition guards* are boolean expressions, which constrain transition firings. A transition guard associated with a transition tests tokens that pass through a transition, and will only enable the transition firings if the guard is successfully matched (i.e., the test evaluates to true). The CP-net transition guards, together with places color sets and arc expressions, appear as a part of net *inscriptions* in the CP-net.

In order to visualize and manage the complexity of large CP-nets, hierarchical CP-nets [Huber *et al.*, 1991, Jensen, 1997a] allow hierarchical representations of CP-nets, in which sub-CP nets can be re-used in higher-level CP nets, or abstracted away from them. Hierarchical CP-nets are built from pages, which are themselves CP-nets. *Superpages* present a higher level of hierarchy, and are CP-nets that refer to *subpages*, in addition to transitions and places. A subpage may also function as a superpage to other subpages. This way, multiple hierarchy levels can be used in a hierarchical CP-net structure.

The relationship between a superpage and a subpage is defined by a *substitution transition*, which substitutes a corresponding *subpage instance* on the CP-net superpage structure as a transition in the superpage. The substitution transition *hierarchy inscription* supplies the exact mapping of the superpage places connected to the substitution transition (called *socket nodes*), to the subpage places (called *port nodes*). The *port types* determine the characteristics of the socket node to port node mappings. A complete CP-net hierarchical structure is presented using a *page hierarchy graph*, a directed graph where vertices correspond to pages, and directed edges correspond to direct superpage-subpage relationships.

Timed CP-nets [Jensen, 1997b] extend CP-nets to support the representation of temporal aspects using a *global clock*. Timed CP-net tokens have an additional color attribute called *time stamp*, which refers to the earliest time at which the token may be used. Time stamps can be used by arc expression and transition guards, to enable a timed-transition if and only if it satisfies two conditions: (i) the transition is *color enabled*, i.e. it satisfies the constraints defined by arc expression and transition guards; and (ii) the tokens are *ready*, i.e. the time of the global clock is equal to or greater than the tokens' time stamps. Only then can the transition fire.

Bibliography

[Aiello *et al.*, 2001] M. Aiello, P. Busetta, A. Dona, and L. Serafini. Ontological overhearing. In *Proceedings of ATAL-01*, 2001.

[AUML site, 2005] AUML site. Agent unified modelling language, at *www.auml.org*, 2005.

[Avrahami and Kaminka, 2005] D. Avrahami and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of IJCAI-05*, 2005.

[Best *et al.*, 2003] R. Best, G. de Valence, and C.A. Langston. *Workplace Strategies and Facilities Management*. Elsevier, 2003.

[Busetta *et al.*, 2001] P. Busetta, L. Serafini, D. Singh, and F. Zini. Extending multi-agent cooperation by overhearing. In *Proceedings of CoopIS-01*, 2001.

[Busetta *et al.*, 2002] P. Busetta, A. Dona, and M. Nori. Channelled multicast for group communications. In *Proceedings of AAMAS-02*, 2002.

[Carrbery, 2001] S. Carrbery. Techniques for plan recognition. *User Modelling and User-Adapted Interaction*, 11:31–48, 2001.

[ChaibDraa, 2002] B. ChaibDraa. Trends in agent communication languages. *Computational Intelligence*, 18(2):89–101, 2002.

[Charniak and Goldman, 1993] E. Charniak and R.P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence Journal (AIJ)*, 64(1):53–79, 1993.

[Cohen and Levesque, 1991] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[Corkill and Lander, 1998] D.D. Corkill and S.E. Lander. Diversity in agent organizations. *Object Magazine*, 8(4):41–47, 1998.

[Cost *et al.*, 1999] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modelling agent conversations with colored Petri nets. In *Proceedings of the Workshop on Specifying and Implementing Conversation Policies, the Third International Conference on Autonomous Agents (Agents-99)*, Seattle, Washington, 1999.

[Cost *et al.*, 2000] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Using colored petri nets for a conversation modelling. In F. Dignum and M Greaves, editors, *Issues in Agent Communications, Lecture notes in Computer Science*, pages 178–192. Springer-Verlag, 2000.

[Cost, 1999] R. S. Cost. *A framework for developing conversational agents*. PhD thesis, Department of Computer Science, University of Maryland, 1999.

[Cranefield *et al.*, 2002] S. Cranefield, M. Purvis, M. Nowostawski, and P. Hwang. Ontologies for interaction protocols. In *Proceedings of the Workshop on Ontologies in Agent Systems, the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-02)*, Bologna, Italy, 2002.

[de Silva *et al.*, 2003] L. P. de Silva, M. Winikoff, and W. Liu. Extending agents by transmitting protocols in open systems. In *Proceedings of the Workshop on Challenges in Open Agent Systems, the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-03)*, Melbourne, Australia, 2003.

[Desel *et al.*, 1997] J. Desel, A. Oberweis, and T. Zimmer. Validation of information system models: Petri nets and test case generation. In *Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics: Computational Cybernetics and Simulation*, pages 3401–3406, Orlando, Florida, 1997.

[Dewan *et al.*, 1997] R.M. Dewan, A. Seidmann, and S. Sundaresan. Communications in hierarchical organizations and standards policies for information technology. *International Journal of Electronic Commerce*, 1(3):43–64, 1997.

[Dignum and Vreeswijk, 2003] F. Dignum and G. Vreeswijk. Towards a testbed from multi-party dialogues. In *Workshop on Agent Communication Languages, LNAI 2292*, pages 212–230, 2003.

[Dignum, 2003] V. Dignum. A model for organizational interaction: Based on agents, founded in logic. SIKS Dissertation Series, 2003.

[Fan and Yen, 2005] X. Fan and J. Yen. Conversation pattern-based anticipation of teammates information needs via overhearing. In *Proceedings of the IEEE/WIC Intelligent Agent Technology conference (IAT-05)*, page 316322, France, 2005.

[Finin *et al.*, 1997] T. Finin, Y. Labrou, and J Mayfield. KQML as an agent communication language. In J Bradshaw, editor, *Software Agents*. MIT Press, 1997.

[FIPA Communicative Acts, 2005] FIPA Communicative Acts. Fipa Communicative Act Library Specification, version J, at *www.fipa.org/specs/fipa0000037*, 2005.

[FIPA site, 2005] FIPA site. Fipa - the Foundation for Intelligent Physical Agents, at *www.fipa.org*, 2005.

[FIPA Specifications, 2005a] FIPA Specifications. Fipa Brokering Interaction Protocol Specification, version H, at *www.fipa.org/specs/fipa0000033/*, 2005.

[FIPA Specifications, 2005b] FIPA Specifications. Fipa Contract Net Interaction Protocol Specification, version H, at *www.fipa.org/specs/fipa0000029/*, 2005.

[FIPA Specifications, 2005c] FIPA Specifications. Fipa Interaction Protocol Library Specification, version E, at *www.fipa.org/specs/fipa0000025/*, 2005.

[FIPA Specifications, 2005d] FIPA Specifications. Fipa Query Interaction Protocol Specification, version H, at *www.fipa.org/specs/fipa0000027/*, 2005.

[FIPA Specifications, 2005e] FIPA Specifications. Fipa Request Interaction Protocol Specification, version H, at *www.fipa.org/specs/fipa0000026/*, 2005.

[Friebel and Raith, 2004] G. Friebel and M. Raith. Abuse of authority and hierarchical communications. *RAND Journal of Economics*, 35(2):224–244, 2004.

[Friedell, 1967] M. F. Friedell. Organizations as semilattices. *American Sociological Review*, 32:46–54, 1967.

[Gannon and Newman, 2001] M. J Gannon and K.L. Newman, editors. *The Blackwell Handbook of Cross-Cultural Management*. Blackwell Publishing, 2001.

[Geib and Harp, 2004] C. W. Geib and S. A. Harp. Empirical analysis of a probalistic task tracking algorithm. In *Proceedings of AAMAS-04 Workshop on Modelling Others from Observations (MOO-04)*, 2004.

[Grossi *et al.*, 2005] D. Grossi, F. Dignum, L. Royakkers, and M. Dastani. Foundations of organizational structures in multi-agent systems. In *Proceedings of AAMAS-05*, 2005.

[Grosz and Kraus, 1996] B. J. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

[Guessoum *et al.*, 2004] Z. Guessoum, M. Ziane, and N. Faci. Monitoring and organizational-level adaptation of multi-agent systems. In *Proceedings of AAMAS-04*, 2004.

[Hameurlain, 2003] N. Hameurlain. MIP-Nets: Refinement of open protocols for modelling and analysis of complex interactions in multi-agent systems. In *Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS-03)*, pages 423–434, Prague, Czech Republic, 2003.

[Horling and Lesser, 2004] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. Computer Science Technical Report 04-45, University of Massachusetts, 2004.

[Horty and Pollack, 2001] J. Horty and M. Pollack. Evaluating new options in context of existing plans. *Artificial Intelligence*, 127(2):199–220, 2001.

[Huber *et al.*, 1991] P. Huber, K. Jensen, and R. M. Shapiro. Hierarchies in Coloured Petri nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets: Theory and Application*, pages 215–243. Springer-Verlag, 1991.

[Jennings, 1993] N. R. Jennings. Commitments and conventions: The foundations of coordination in multi-agent systems. *Knowledge Engineering Review*, 8(3):223–250, 1993.

[Jensen, 1997a] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer-Verlag, 1997.

[Jensen, 1997b] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 2. Springer-Verlag, 1997.

[Jensen, 1997c] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 3. Springer-Verlag, 1997.

[Jensen, 2003] M.T. Jensen. Organizational communication: A review. R&D Report 1/2003, Agderforskning, Norway, 2003.

[Kalech and Kaminka, 2005] M. Kalech and G. A. Kaminka. Towards model-based diagnosis of coordination failures. In *Proceedings of AAAI-05*, 2005.

[Kaminka and Tambe, 2000] G. A. Kaminka and M. Tambe. Robust multi-agent teams via socially-attentive monitoring. *JAIR*, 12:105–147, 2000.

[Kaminka *et al.*, 2002] G.A. Kaminka, D.V. Pynadath, and M. Tambe. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *JAIR*, 17:83–135, 2002.

[Kaminka, 2000] G. A. Kaminka. *Execution Monitoring in Multi-Agent Environments*. PhD thesis, University of Southern California, Computer Science Department, 2000.

[Kautz and Allen, 1986] H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37. AAAI Press, 1986.

[Khomenco and Koutny, 2000] V. Khomenco and M. Koutny. LP deadlock checking using partial order dependencies. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR-00)*, pages 410–425, Pennsylvania State University, Pennsylvania, 2000.

[Klein and Dellarocas, 1999] M. Klein and C. Dellarocas. Exception handling in agent systems. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, 1999.

[Klein, 1997] G. A. Klein. The recognition-primed decision (rpd) model: Looking back, looking forward. In C. E. Szambok and G. Klein, editors, *Naturalistic decision making*, pages 285–292. 1997.

[Kone *et al.*, 2000] M. T. Kone, A. Shimazu, and T. Nakajima. The state of the art in agent communication languages. *Knowledge and Information Systems*, 2:258–284, 2000.

[Kumar *et al.*, 2000] S. Kumar, M.J. Huber, D.R. McGee, P.R. Cohen, and H.J. Levesque. Semantics of agent communication languages for group interaction. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 42–47. AAAI Press/The MIT Press, 2000.

[Legras, 2002] F. Legras. Using overhearing for local group formation. In *Proceedings of AAMAS-02*, 2002.

[Lin *et al.*, 2000] F. Lin, D. H. Norrie, W. Shen, and R. Kremer. A schema-based approach to specifying conversation policies. In F. Dignum and M Greaves, editors, *Issues in Agent Communications, Lecture notes in Computer Science*, pages 193–204. Springer-Verlag, 2000.

[Ling and Loke, 2003] S. Ling and S. W. Loke. MIP-Nets: A compositional model of multi-agent interaction. In *Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS-03)*, pages 61–72, Prague, Czech Republic, 2003.

[Mazouzi *et al.*, 2002] H. Mazouzi, A. E. Fallah-Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-02)*, pages 517–526, Bologna, Italy, 2002.

[McElhearn, 1996] K. McElhearn. Writing conversation: an analysis of speech events in e-mail mailing lists, 1996.

[Milner *et al.*, 1990] R. Milner, R. Harper, and M. Tofte. *The Definition of Standard ML*. MIT Press, 1990.

[Moldt and Wienberg, 1997] D. Moldt and F. Wienberg. Multi-agent systems based on Coloured Petri nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets (ICATPN-97)*, pages 82–101, Toulouse, France, 1997.

[Morgenstern, 1951] O. Morgenstern. Prolegomena to a theory of organizations. Manuscript., 1951.

[Ndumu *et al.*, 1999] D. Ndumu, H. Nwana, L. Lee, and J. Collins. Visualizing and debugging distributed multi-agent systems. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 326–333, 1999.

[Novick and Ward, 1993] D.G. Novick and K. Ward. Mutual beliefs of multiple conversants: A computational model of collaboration in air traffic control. In *Proceedings of AAAI-93*, pages 196–201, 1993.

[Nowostawski *et al.*, 2001] M. Nowostawski, M. Purvis, and S. Cranefield. A layered approach for modeling agent conversations. In *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS and Scalable MAS, the Fifth International Conference on Autonomous Agents*, pages 163–170, Montreal, Canada, 2001.

[Odell *et al.*, 2000] J. Odell, H. V. D. Parunak, and B. Bauer. Extending UML in the design of multi-agent systems. In *Proceedings of the AAAI-2000 Workshop on Agent-Oriented Information Systems (AOIS-00)*, 2000.

[Odell *et al.*, 2001a] J. Odell, H. V. D. Parunak, and B. Bauer. Agent UML: A formalism for specifying multi-agent interactions. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer-Verlag, Berlin, 2001.

[Odell *et al.*, 2001b] J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in UML. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 121–140. Springer-Verlag, Berlin, 2001.

[Packel *et al.*, 1992] E. W. Packel, J. F. Traub, and H. Wozniakowski. Measures of uncertainty and information in computation. *Information Sciences: an International Journal*, 65(3):253–273, 1992.

[Parker, 1993] L. E. Parker. Designing control laws for cooperative agent teams. In *Proceedings of the IEEE Robotics and Automation Conference*, pages 582–587, Atlanta, GA, 1993.

[Paruchuri *et al.*, 2004] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Towards a formalization of teamwork with resource constraints. In *Proceedings of AAMAS*, pages 596–603, 2004.

[Parunak, 1996] H. V. D. Parunak. Visualizing agent conversations: Using enhances Dooley graphs for agent design and analysis. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, 1996.

[Paurobally and Cunningham, 2003] S. Paurobally and J. Cunningham. Achieving common interaction protocols in open agent environments. In *Proceedings of the Workshop on Challenges in Open Agent Systems, the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-03)*, Melbourne, Australia, 2003.

[Paurobally *et al.*, 2003] S. Paurobally, J. Cunningham, and N. R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proceedings of the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-03)*, Melbourne, Australia, 2003.

[Petri Nets site, 2005] Petri Nets site. Petri nets world: Online services for the international petri nets community, at *www.daimi.au.dk/petrinets*, 2005.

[Platon *et al.*, 2004] E. Platon, N. Sabouret, and S. Honiden. T-compound: An agent-specific design pattern and its environment. In *Proceedings of 3rd international workshop on Agent Oriented Methodologies at OOPSLA 2004*, pages 63–74, 2004.

[Platon *et al.*, 2005] E. Platon, N. Sabouret, and S. Honiden. Overhearing and direct interactions: Point of view of an active environment, a preliminary study. In *Proceedings of AAMAS-05 Workshop on Environment for Multi-Agent Systems*, 2005.

[Poutakidis *et al.*, 2002] D. Poutakidis, L. Padgham, and M. Winikoff. Debugging multi-agent systems using design artifacts: The case of interaction protocols. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-02)*, pages 960–967, Bologna, Italy, 2002.

[Purvis *et al.*, 2002] M. K. Purvis, P. Hwang, M. A. Purvis, S. J. Cranefield, and M. Schievink. Interaction protocols for a network of environmental problem solvers. In *Proceedings of the 2002 iEMSs International Meeting:Integrated Assessment and Decision Support (iEMSs 2002)*, pages 318–323, Lugano, Switzerland, 2002.

[Pynadath and Tambe, 2003] D. V. Pynadath and M. Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):71–100, 2003.

[Ramos *et al.*, 2002] F. Ramos, J. Frausto, and F. Camargo. A methodology for modeling interactions in cooperative information systems using Coloured Petri nets. *International Journal of Software Engineering and Knowledge Engineering*, 12(6):619–636, 2002.

[Reisig, 1985] W. Reisig. *Petri Nets: An Introduction.* Springer-Verlag, 1985.

[Rossi and Busetta, 2004] Silvia Rossi and Paulo Busetta. Towards monitoring of group interactions and social roles via overhearing. In *Proceedings of CIA-04*, pages 47–61, Erfurt, Germany, 2004.

[Rossi and Busetta, 2005] S. Rossi and P. Busetta. With a little help from a friend: Applying overhearing to teamwork. In *Proceedings of IJCAI-05 Workshop on Modelling Others from Observations (MOO)*, 2005.

[Scerri *et al.*, 2004] P. Scerri, Yang. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *Proceedings of AAMAS-04*, 2004.

[Selznick, 1948] P. Selznick. Foundations of the theory of organizations. *American Sociological Review*, 13:25–35, 1948.

[Smith and Cohen, 1996] I. A. Smith and P. R. Cohen. Toward a semantics for an agent communications language based on speech-acts. In *Proceedings of AAAI-96*, 1996.

[So and Durfee, 1996] Y. So and E. Durfee. Designing tree-structured organizations for computational agents. *Computational and Mathematical Organization Theory*, 2(3):219–246, 1996.

[Tambe, 1997] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.

[Traub and Werschulz, 1998] J. Traub and A. G. Werschulz. *Complexity and Information.* Cambridge University Press, 1998.

[Traum and Rickel, 2002] D. Traum and J. Rickel. Embodied agents for multi-party dialogues in immerse virtual worlds. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS-02)*, pages 766–773, Bologna, Italy, 2002.

[Turner and Jennings, 2000] P. J. Turner and N. R. Jennings. Improving the scalability of multi-agent systems. In *Proceedings of the First International Workshop on Infrastructure for Scalable Multi-Agent Systems*, Barcelona, Spain, 2000.

[Wikstrom, 1987] A. Wikstrom. *Functional Programming using Standard ML.* International Series in Computer Science. Prentice-Hall, 1987.

[Wilkins *et al.*, 2003] D. E. Wilkins, T. J. Lee, and P. Berry. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18:217–261, 2003.

[Xu and Shatz, 2001] H. Xu and S. M. Shatz. An agent-based Petri net model with application to seller/buyer design in electronic commerce. In *Proceedings of the 5th International Symposium on Autonomous Decentralized Systems (ISAD-01)*, pages 11–18, Dallas, Texas, USA, 2001.

[Yadgar *et al.*, 2003] O. Yadgar, S. Kraus, and O. Ortiz. Hierarchical information combination in large-scale multi-agent resource management. In M.P. Huget, editor, *Communication in MAS: Background, current trends and future*, pages 129–145. Springer-Verlag, 2003.