2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

A Colored Petri Net Based Approach for Multi-agent Interactions

Quan Bai, Minjie Zhang and Khin Than Win
School of IT and Computer Science,
University of Wollongong, NSW, Australia
{qb92, minjie, win}@uow.edu.au

Abstract

In a multi-agent system (MAS), agent interactions are established through exchanging messages following interaction protocols. Most current agent interaction protocols are hard-coded within agents during the agent designing, and this feature greatly reduces the flexibility of the agent interactions. In this paper, we present a Colored Petri Nets (CPNs) based approach to form flexible interaction protocols between agents. Through this approach, agents can analyse a protocol and check whether the protocol is "understandable" and advantageous to the objectives of agents.

Keywords: multi-agent system, agent interaction, interaction protocol, Colored Petri Net

1 Introduction

Multi-agent systems (MASs) are computational systems in which two or more agents interact or work together to perform a set of tasks or to satisfy a set of goals [1]. Agents of a multi-agent system (MAS) need to interact with others toward their common objective or individual benefits of themselves. Agent interactions are established through exchanging messages that specify the desired performatives of other agents (such as notice, request, etc.) and declarative representations of the contents of messages. In general, messages exchanged among agents are composed in agent communication languages (ACLs), such as Knowledge Query and Manipulation Language (KQML) [2] and the Foundation for Intelligent Physical Agents (FIPA) ACL [3]. In addition, messages exchanged between agents need to follow some standard patterns, which are described in agent interaction protocols [4]. Interaction protocols constrain the possible sequences of messages that may occur in agent interactions. An agent needs to indicate the interaction protocol, which it wishes to follow, to others before it starts the interactions with them, and then the recipients will follow the pattern described in the protocol to start the conversation if they accept the protocol.

As the application domains of MASs are getting more and more complex, many current agent interaction protocols appear some limitations that impede MAS implementations. Firstly, many current application domains of MASs require agents to work in changing and uncertain environments (open environments). In such environments, interactions between agents may be influenced by some unexpected factors, such as unexpected message, loss of messages or deviation in the message order. Most current agent interaction protocols are lack of mechanisms to handle these unexpected factors. Secondly, agent architectures in some MASs are heterogeneous and different agents may possess different interaction protocols. Therefore, due to the heterogeneity, when an agent initialises an interaction with others, it cannot guaranty that its interaction protocol can be understood and accepted by other agents. Thirdly, most agents are hard-coded with interaction protocols, which means problems, such as when to use a particular protocol, what information to transmit, what order to execute tasks, etc., are left to agent designers. This feature reduces the flexibility of the agent interactions because protocols are hard to be modified at runtime once they are pre-coded into the agents. Finally, many current interaction protocols, such as KQML, are not designed particularly to carry knowledge. This kind of knowledge "poor" [6] protocols are not suitable for applications that need to exchange complex knowledge. In other words, lack of flexibility and robustness of many current interaction protocols greatly limits the implementation of MASs, and how to build a flexible and knowledge "rich" interaction protocol has become one of the main research issues in the area of MASs.

Colored Petri Nets (CPNs) [5] provide an appropriate mathematical formalism for the description, construction and analysis of distributed and concurrent systems. CPNs can express a great range of interactions in graphical representations and well-defined semantics, and allow formal analysis and transformations [5]. There are a number of works of using Petri Nets (PNs) or CPNs to model agent interaction protocols. Mariusz describes a layered approach based on CPNs that can be used for modelling complex, concurrent conversations among

agents in a multi-agent system [7]. Cost proposes the use of CPNs as a model underlying a language for protocol specification by taking the advantages of CPNs' great expressive power to support for concurrency [8]. Poutakidis uses interaction protocols to debug agent interaction, and uses the Petri Nets to monitor conversations and to provide precise and informative error messages when protocols are not correctly followed by the agents [9]. There have been also some works on the investigation of protocols' flexibility, robustness and extensibility. Hutchison gives an example protocol, called Merchant-Customer protocol, to describe flexibility and robustness of agent interaction protocols in open systems [10]. Ahn suggests a handshaking mechanism for conversation policy agreements that enables agents to exchange and agree to new conversation policies at runtime [11]. Freire describes an approach that enables agents to execute any interaction protocol that can be expressed in the proposed XML representation [12]. To cover some limitations of current agent interaction protocols, in this paper, we present a Colored Petri Net (CPN) based approach for agent interaction. In this approach, agents may not have predefined interaction protocols, and they can form interaction protocols dynamically. An agent can analyse a protocol and check whether the protocol is suitable for itself, then makes a decision about whether it follows the protocol or not.

The remainder of the paper is arranged as follows. The detail description about PNs, CPNs, and how to use CPNs to model agent protocol is presented in Section 2. The CPN based approach to form and evaluate a flexible protocol between agents is introduced in Section 3. Finally, conclusions and future direction of this work are presented in Section 4.

2 PNs, CPNs and Using CPNs to Model Interaction Protocols

PNs and CPNs provide a framework for the construction and analysis of distributed and concurrent systems. A CPN model of a system describes the states, which the system may be in, and the transitions between these states. A brief description about PNs and CPNs is presented in this section.

2.1 PNs and CPNs

The basic structure of a PN can be formally defined by a 4-tuple (P, T, A, N) (see Figure 1), where *P* is a set of *Places*, such as P1, P2, P3 and P4; *T* is a set of *Transitions*, such as T1, T2 and T3; *A* is a set of *Arcs*, such as the arc from P1 to T1, T1 to P1, P2 to T1, etc.; and *N* is a set of *Token*, for example, in Figure 1, P1 and P3 have one token in the initial state. Net structure and transition firing rules are associated together to describe how system states transfer. There are a number of transition firing rules associated with different types of PNs. However, all kinds PNs share a common firing property: a transition can be fired if the token number of all input places is equal or greater than their arcs' weights [13]. After a transition is fired, the tokens of its input places will be moved to its output places.

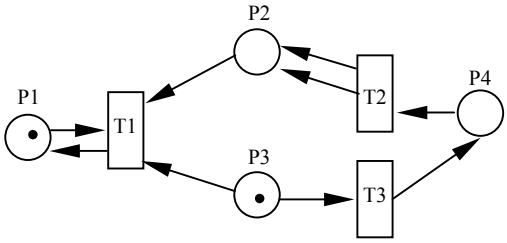

Figure1: An example of PNs

A CPN can be defined by a 9-tuple (Σ, P, T, A, N, C, G, E, I), where Σ is a set of non-empty types, also called Colored sets; *P* is a set of Places; *T* is a set of Transitions; *A* is a set of Arcs; *N* is a node function; *C* is a color function; *G* is a guard function; *E* is an arc expression function; and *I* is a an initialization function.

CPNs differ from PNs because their tokens are not simply blank markers, but have data associated with them. A token's *color* is a schema or specification. Places of CPNs contain *multi-sets* of tokens. Arcs of CPNs specify the token/tokens that they can carry and can also specify some transfer conditions. Arcs exiting and entering a place can have an associated constrain function to determine which multi-set elements are to removed or hold. Transitions of CPNs are associated with some guard functions that enforce some constraints on tokens.

2.2 Use CPNs to Model Agent Interaction Protocols

It is becoming consensus that a CPN is one of the best ways to model agent interaction protocols [8, 4, 7, 9]. By using CPNs, an agent interaction protocol can be modeled as a net of components, which are carriers of the protocol structure and the interaction policy.

Using CPNs to model agent interaction protocol, the states of an agent interaction are represented by CPN places. Each place has an associated type determining the kind of data that the place may contain. Data exchanged between agents are represented by tokens, and the colors of tokens indicate the data value of the tokens. The interaction policies of a protocol are carried by CPN transitions and their associated arcs. A transition is enabled if all of its input places have tokens, and the colors of these tokens can satisfy constraints that are specified on the arcs. A transition can be *fired*, which means the actions of this transition can occur, when this transition is enabled.

When a transition occurs, it consumes all the input tokens as computing parameters, conducts conversation policy and adds new tokens into all of its output places. After a transition occurs, the state (marking) of a protocol has been changed and a protocol will be in terminal state when there is no enabled or fired transition.

Here, we take FIPA Inform Protocol [3] as an example to indicate how to use CPNs to model agent interaction protocols. The FIPA Request protocol can be modeled as a CPN, which is shown in Figure 2. From Figure 2, we can see that there are five states in the interaction and these states are represented in five places, respectively. If there are one or more than one token in "Start" place and these tokens can satisfy the constraints that are specified in "send" arc (belongs to data type "message"), the "inform" transition will be enabled. After "inform" transition is fired, a token will be removed from "Start" place, and "Received" and "Terminated1" place will get a new token, respectively. After "Receive" place get a token, "process inform" transition will be enabled. Finally, the interaction will be terminated after process inform transition is fired.

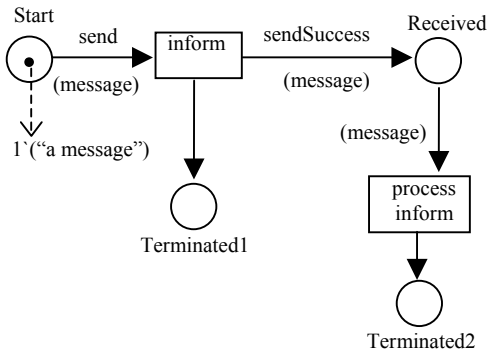

Figure 2: Use CPN to represent FIPA inform protocol

3 CPN Based Flexible Protocol Forming Approach

As we mentioned in the first section, most agents are hard-coded with interaction protocols, and this reduces the flexibility of the agent interactions because protocols are hard to be modified at runtime once they are pre-coded into the agents. In this section, we present a CPN based approach that can enable agents to form interactions flexibly. In the implementation of this approach, agents do not need to interact with other agents with a fixed protocol. Agents' interaction protocols will be generated during interactions, and they also can modify their protocols according to their status during the interactions. In general, the approach is composed by two main procedures, which are sending protocol specifications and protocol analysis.

3.1 The Default Protocol and Sending Protocol Specifications

In this approach, agents of the system have a default interaction protocol, which is shown in Figure 3. The default protocol is composed by three components: two places that belong to Protocol Specification (PS) type, and one transition. The related description of the default interaction protocol is described as following.

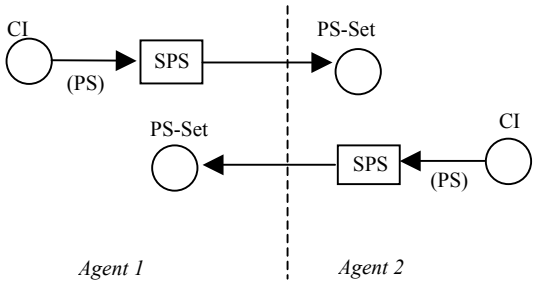

Figure 3: Default interaction protocol

PS (Protocol Specification): PS is a data type of the approach. A PS token contains a protocol specification that indicates an interaction protocol. The format of the specification can be referred to Figure 4.

CI (Call Interaction) Place: in the default protocol, a CI place is the place that conducts PS tokens. Agent may put a PS token, which specifies its desired interaction protocol, into a CI Place when the agent needs to interact with some other agent(s).

PS-Set (Protocol Specification Set) Place: in the default protocol, a PS-set place is the place that receives PS tokens, which are sent by other agents.

SPS (Send Protocol Specification) Transition: in the default protocol, a SPS transition sends Protocol Specifications (PSs). A CI place and PS-Set place are the input and output places of SPS transition, respectively. The SPS transition can be enabled when there is/are token(s) in its CI place. A PS token will be removed from the CI place to the PS-Set place after the SPS transition fires once.

```
/* Interaction Protocol
   Description: agent inform protocol */
(protocol
   (name inform)
   (place (name messagePlace) (owner sender) (PT message) )
   (place (name receivePlace) (owner receiver) (PT message) )
   (place (name failPlace) (owner sender) (PT error_message) )
   (transition (name sendMessage) (arc (name send)
         (from messagePlace) (match message) ) (arc (name
         sendSuccess) (to receivePlace) (match message) )
         (arc (name sendFail) (to failPlace) (match
         error_message) ) )
```

Figure 4: Specification of inform protocol (an example)

The first procedure of the approach is Sending Protocol Specifications. When an agent needs to interact with some other agent, it composes a PS to describe its desired interaction protocol according to its requirement, and puts the PS into its CI place. The content of PS is a specification of a CPN modeled interaction representation. In this specification, the agent can indicate its required data as a place that needs the other agent to input token. The data type constraints can be described in corresponding place types, and the data value constraints (token colors) can be described in constraint functions of arcs. The format of the specification can be as the format of Figure 4 or defined by users. The corresponding CPN model of Figure 4 is shown in Figure 5.

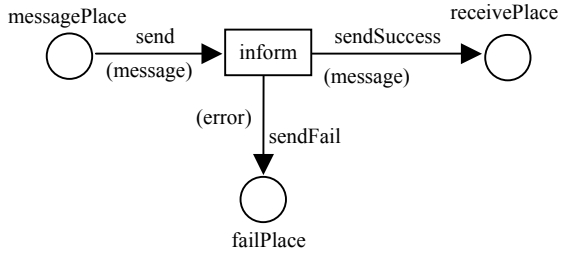

Figure 5: CPN model of inform protocol

3.2 Protocol Analysis

After an agent receives a protocol specification, it will analyse whether it can execute the interaction with its current status. In PN theory [13], there are many methods to analyze whether a PN model is executable. In this paper, we use Matrix Equation methods [13] to evaluate interaction protocols that an agent received. Before introducing the protocol analysis method, we first give some related definitions.

Definition 1: the Place Type (PT) of a place is the associated data type that determines the kind of data/tokens, which the place can contain.

Definition 2: when an agent receives a PS, the PS Place Type Set (PTS) is the set of place types that occur in the PS. For example, the PTS of the PS in Figure 5 is {message, error_message}.

Definition 3: the Understandable Type Set (UTS) of an agent is the set of data types that exist in the knowledge base of the agent.

3.2.1 Place Type Analysis

The first step of Protocol Analysis (Place Type Analysis) is to test whether the agent can understand the PS that it received.

Definition 4: If an agent with UTS receives a PS, the Non-understandable Type Set (NTS) of the agent is: NTS = PTS ─ (PTS ∩ UTS), where PTS is the PS Place Type Set of the PS and UTS is the Understandable Type Set of the agent.

Definition 5: An agent can understand a PS when PTS ⊆ UTS or NTS = ∅, where PTS is the PS Place Type Set of the PS, UTS is the Understandable Type Set of the agent and NTS is the Non-understandable Type Set of the agent.

During Place Type Analysis, the agent generates the NTS according to its UTS and the PTS of the PS token. If the agent can understand the PS, which means that the generated UTS is empty, the protocol analysis will process the second step. Otherwise, the agent will attach its comments, which indicate the place type that it does not understand, to the PS and send the PS back to the sender. Here, we give a simple example of Place Type Analysis in Figure 6. In Figure 6, there are two PS tokens in the PS-Set place (also refer to Figure 3) of Agent 3. These two PS tokens are P1 and P2, which are received from Agent 1 and Agent 2, respectively. The table on the right side of the figure lists the UTS of Agent 3. Comparing P1 and P2 with the UTS of Agent 3, we can see that Agent 3 can understand P1 but not P2 because the DataType7 in P2 is not in the UTS of Agent 3. Therefore, Agent 3 will comment, "DataType7 is un-understandable", on P2, and send P2 back to Agent 2.

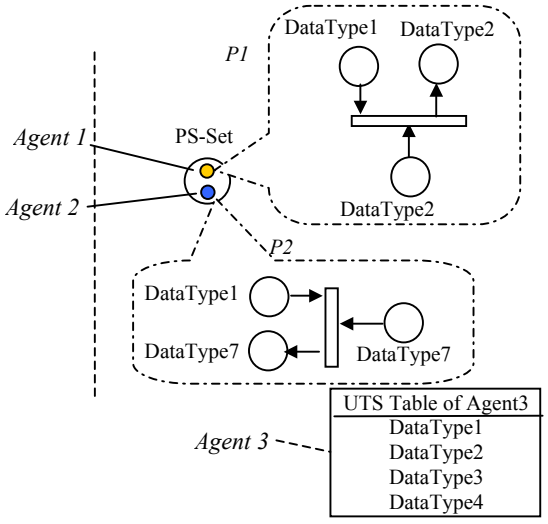

Figure 6: A simple example for Place Type Analysis

3.2.2 Interaction Analysis

The second step of Protocol Analysis (State Checking) is to test whether the current status of the agent is satisfactory to accept the PS and whether the interaction will be conflict with the goal of the agent. According to *Matrix Equation Method* [13] of PN theory, a PN model can be expressed in a matrix format. For instance, the matrix format of the PN model of Figure 1 can be described as Equation (1). In Equation (1), D^+ and D^- are matrices to represent input and output functions of the PN model, respectively.

$$D = D^+ - D^- = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & +1 & -1 \\ 0 & 0 & -1 & +1 \end{bmatrix} \quad (1)$$

The marking of a PN model can be represented as Marking Set (m_1, m_2, …, m_n), where m_n represents the token number of nth place. For example, the marking of Figure 1 can be represented as (1, 0, 1, 0). Toward the interaction analysis problems, we give following definitions, which are based on the Matrix Equation Method.

Definition 6: *IM (Interaction Matrix)* is the matrix of an interaction protocol PN model. IM^- and IM^+ are matrices to represent input and output functions of the interaction protocol PN model. $IM = IM^+ - IM^-$.

Definition 7: the *Required Token Set (RTS)* of a PS is the *multi-set* of tokens that the PS requires the agent to offer.

Definition 8: the *Gain Token Set (GTS)* of a PS is the *multi-set* [14] of tokens that the agent gets through interactions.

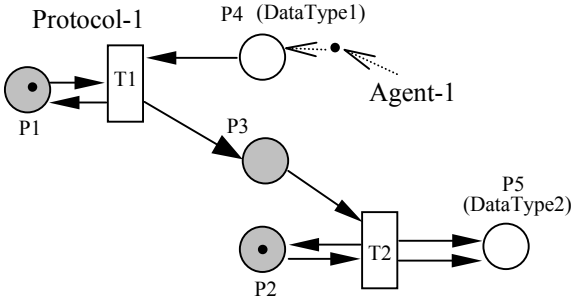

Figure 7. An example of Gain Analysis

With the IM of an interaction protocol, an agent can calculate the GTS that it will gain through the interaction. Furthermore, the agent is able to check whether the perspective result of the interaction is conflict with its own objective. For example, if *Agent-1* received an interaction protocol *Protocol-1*, which is shown in Figure 7 (in Figure 7, the white places need Agent-1 to input/output tokens and grey places are maintained by other agents), the IM of *Protocol-1* is

$$IM = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 2 \end{bmatrix} \quad (2)$$

To make the transitions of *Protocol-1* fire, we need to have a marking $\mu = (1, 1, 0, 1, 0)$. Since P1, P2 and P3 are maintained by other agents, so Agent-1 only needs to input/output tokens in/from P4, and P5. From IM^- we can see that only P4 is an input place, so the protocol only requires Agent-1 to input tokens into P4. Therefore, we can have *RTS* = (0, 0, 0, 1, 0). Furthermore, we can find out the GTS of Protocol-1 through following calculations.

$$\mu' = \mu + x_1 \cdot IE$$
$$= (1,1,0,1,0) + (1,0) \cdot \begin{bmatrix} 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & -1 & 0 & +2 \end{bmatrix}$$
$$= (1,1,1,0,0) \quad (3)$$

$$\mu'' = \mu' + x_2 \cdot IE$$
$$= (1,1,1,0,0) + (0,1) \cdot \begin{bmatrix} 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & -1 & 0 & +2 \end{bmatrix}$$
$$= (1,1,0,0,2) \quad (4)$$

In above calculations, Equation (3) shows the marking transition after T1 is fired, and Equation (4) shows the marking transition after T2 is fired (also see Figure 8). Since Agent-1 can only gain tokens from P5, so the GTS of Protocol-1 is (0, 0, 0, 0, 2). According to the Data Type of P4 and P5, we can see that through the interaction, Agent-1 will lose one token of DataType1 and gain two token of DataType2. Therefore, Agent-1 will evaluate whether the interaction is advantageous or harmful to its own goals.

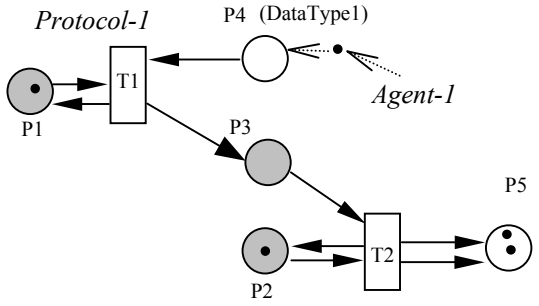

Figure 8. An example of Gain Analysis (after T1 and T2 are fired)

4 Future Works and Conclusions

For MASs, predefined agent interaction protocols reduce the flexibility of agent interaction especially in open environments. In this paper, we have proposed a CPN based approach to enable agents to form interaction protocols flexibly. Furthermore, in this approach, agents can also analyse whether the received protocol is understandable, whether the interaction can be accepted with the current status of the agent and whether the interaction conflicts with the objectives of the agent. Our further work is

targeted at using CPNs to model and represent the status of agents, and establish CPN based mechanism to coordinate agent interactions. This mechanism may include protocol forming strategies, conflict analysis strategies, conflict solving strategies and exception handling strategies.

Acknowledgements

This research was supported by a large grant from the Australian Research Council under contract LX0346249 and Small Grant from University of Wollongong under contract SG04-227381019.

References

[1] Lesser, V., Cooperative Multiagent Systems: A Personal View of the State of the Art, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, NO. 1, pp.133-142 (1999).

[2] Finin, T., Labrou, Y. and Mayfield, J., KQML as an agent communication language, J.M. Bradshaw, editor, Software Agents. MIT Press, pp. 291-316 (1997).

[3] Foundation for Intelligent Physical Agents. FIPA ACL message representation in string specification, http://www.fipa.org/specs/fipa00070/, visited on 18/7/2004.

[4] Cranefield, S., Purvis, M., Nowostawski, M. and Hwang, P., Ontology for Interaction Protocols, *Proceedings of the Second International Workshop on Ontologies in Agent Systems*, Bologna, Italy (2002), http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-66/oas02-16.pdf, visited on 18/7/2004.

[5] Jensen, K. *Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use, volume 1: Basic Concepts*. Springer-Verlag, Berlin (1992).

[6] Lesser, V., Reflections on the Nature of Multi-agent Coordination and Its Implications for an Agent Architecture, *Autonomous Agents and Multi-agent Systems*, Vol. 1, pp 89-111, (1998).

[7] Nowostawski, M., Purvis, M. and Cranefield, S., A Layered Approach for Modelling Agent Conversations, *Proceedings of the 2nd International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, pp.163-170, Montreal Canada, (2001).

[8] Cost, R., Modelling Agent Conversations with Coloured Petri Nets, *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pp.59-66, Seattle, Washington, (1999).

[9] Poutakidis, D., Padgham, L. and Winikoff, M., Debugging Multi-Agent Systems Using Design Artefacts: The Case of Interaction Protocols, In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi Agent Systems,* pp.960-967, Bologna, Italy, (2002).

[10] Hutchison, J. and Winikoff, M., Flexibility and Robustness in Agent Interaction Protocols, *Proceedings of the 1st International Workshop on Challenges in Open Agent Systems*, Bologna, Italy, (2002), http://goanna.cs.rmit.edu.au/~winikoff/Papers/challenge02.pdf, visited on 18/7/2004.

[11] Ahn, H., Lee, H., Yim, H. and Park, S., Handshaking Mechanism for Conversation Policy Agreements in Dynamic Agent Environment, *Proceedings of the 1st International Workshop on Challenges in Open Agent Systems*, Bologna, Italy, (2002), http://www.agentcities.org/Challenge02/Proc/Papers/ch02_9_ahn.pdf, visited on 18/7/2004.

[12] Freire, J. and Botelho, L., Executing Explicitly Represented Protocols, *Proceedings of the 1st International Workshop on Challenges in Open Agent Systems*, Bologna, Italy, (2002), http://www.agentcities.org/Challenge02/Proc/Papers/ch02_15_freire.pdf, visited on 18/7/2004.

[13] Peterson, J., *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Inc., N.J. (1981).

[14] Jensen, K., An Introduction to the Practical Use of Coloured Petri Nets, *Lectures on Petri Nets II: Applications*, Lecture Notes in Computer Science Vol. 1492, Springer-Verlag, pp. 237-292 (1998).