Formalization and Pre-Validation for Interaction Protocols in Multiagent Systems

Jean-Luc Koning, Guillaume François, Yves Demazeau

Formalization and Pre-Validation for Interaction Protocols in Multiagent Systems

Jean-Luc Koning, Guillaume François, Yves Demazeau

Formalization and Pre-Validation for Interaction Protocols in Multiagent Systems

Jean-Luc Koning[1], Guillaume François[2], Yves Demazeau[3]

Abstract. In this article, we take up validation techniques used so far in the domain of communication protocols in networks and adapt a similar method for the validation of interaction protocols in multiagent systems. First of all, we rely on Petri nets in order to model such protocols. Second, we move to a protocol validation stage through the analysis of a Petri net model. We also give a very brief presentation of a software tool that enabled us on one hand to run the analysis, and on the other hand to simulate the unfolding of typical scenarios for sequences of exchanged messages. Finally, we extract the global protocol properties and infer consequences on its pre-validity. The phrase *pre-validity* is used in this paper instead of validity since one is not interested in proving a complete validity of the negotiation protocol but rather a partial one, i.e., checking the satisfaction of some elementary properties.

1 Introduction

Multiagent Systems (MAS) are concerned with coordinating behavior among a collection of autonomous intelligent agents. Such software agents are designed to reconcile their own interests with the constraints implied by the other agents. The main justifications for making use of a multiagent model comes from its distribution, openness, flexibility and robustness features [10].

In this setting, a negotiation process translates into interactions among agents. Interaction is more than an exchange of messages. Issues associated with it, are: *models of agents* (beliefs, goals, representation and reasoning), *interaction protocols* (an interaction regime that guides the agents) and *interaction languages* (languages that introduce standard message types that all agents interpret identically). One way to structure message passing during agent communication is through protocols.

The design of interaction protocols for multiagent systems has already been tackled in the literature like in [1]. These references usually discuss low level communication issues or tools for designing interaction protocols. In the remainder of this article, we will take up validation techniques used so far in the domain of communication protocols in networks [5] and prototyping of distributed systems [7]. We will adapt a similar method for the pre-validation of interaction protocols in multiagent systems. The phrase *pre-validity* is used in this paper instead of validity since one is not interested in proving a complete validity of the negotiation protocol but rather a partial one, i.e., checking the satisfaction of some elementary properties.

First of all, we will rely on Petri nets in order to model such protocols. For lack of space we will briefly introduce related work both in the domain of communication networks and the domain of multiagent systems.

Second, we will move to a protocol validation stage through the analysis of a Petri net model. We will also give a very brief presentation of a software tool that enabled us on one hand to run the analysis, and on the other hand to simulate the unfolding of typical scenarios for sequences of exchanged messages.

Finally, we will extract the global protocol properties and we will infer consequences on its (pre-)validity.

[1] LEIBNIZ-INPG, Grenoble, France
[2] CELAR, Issy-les-Moulineaux, France
[3] LEIBNIZ-CNRS, Grenoble, France

Submitted to ECAI 98
June 2, 1998

2 Petri Nets and their Use in DAI

Modeling an interaction between two agents of a MAS comes down to building a per-agent Petri sub-net as well as a sub-net for the virtual medium, i.e., modeling the overall service that enable to rout messages between agents.

In [8], Ferber discusses a Petri net modeling of an interaction between two agents. It shows how agent A can ask agent B to provide a service. There are at least two reasons for criticizing that approach.

- The protocol that is described only deals with two agents that cannot play a symmetric role. Their statute is different; one requests and the other answers the request.
- The exchanged messages modeled by tokens in the Petri net represent hypotheses or proposals. When a hypothesis is modified by an agent the original one is lost even though its processing by the agents is not over. This stems from the fact that there is a given number of tokens that circulate in the net: one per agent which conveys its internal state and one and only one for the message that is being processed. As a consequence, this Petri net does not model a simultaneous processing of several proposals derived from one initial hypothesis.

A real interaction protocol provides a detailed description of the possible interactions between n (more than two) agents. But there is no clear approach in the DAI literature for modeling interactions that take place between more than two agents at one time. This is probably due to the fact that interaction between agents in DAI are rather grounded on Speech Acts theory than on distributed systems, which constitutes a serious bias in our point of view. As an example, the cooperative learning protocol given in [12] only deals with interactions among two agents.

Other uses of Petri nets in Distributed Artificial Intelligence include planning. In [6] a formalization of concurrent plans is proposed based on recursive Petri nets which is a suitable formalism for distributed planning. They also tackle there the management of potentially shared and conflicting resources through a recursive model.

3 Petri Nets For Representing Protocols

Modeling interactions between agents in a MAS comes down to building a Petri sub-net per agent and a Petri sub-net that represents the virtual medium. That is what we address in this section.

3.1 Models of Virtual Mediums Linking Communicating Entities

Let us first model a perfect virtual medium linking n communicating entities. In a perfect virtual medium the possible loss of messages is not modeled and the transmission capacity is infinite. Such medium must offer the n entities the capability to send and receive several sorts of messages. Figure 1 exemplifies the modeling of such a medium for 3 entities A, B and C that allows each of them to receive and send two different types of messages.

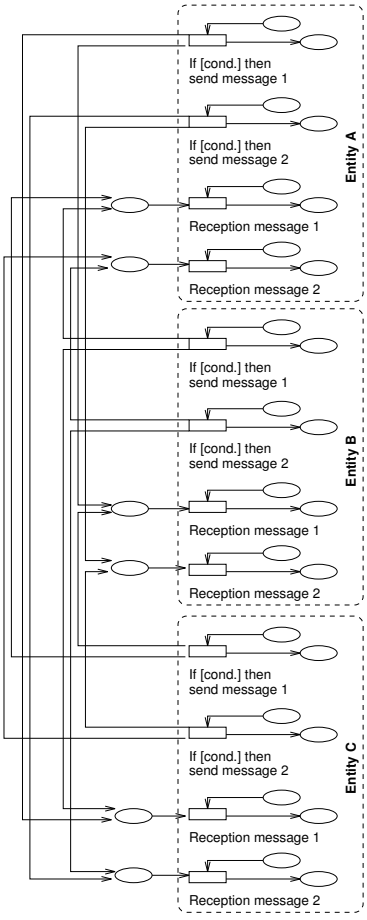

Figure 1. Perfect virtual medium linking three communicating entities, and allowing the exchange of two types of messages.

In a same medium linking n entities and capable of exchanging m different sorts of messages, the number of links would be $n \cdot (n-1) \cdot m$. Clearly this modeling technique is not suitable for a great many entities. The virtual medium's complexity should not depend on the number of communicating entities or at least not be proportional to its square. One could also rely on the bus communication principle, where there is a bus type line of transmission for any kind of message. Figure 2 shows a bus type virtual medium that links three communication entities and enables the exchange of two types of messages.

Box (i, j) carries out the following functions:

- collect j-type messages sent by entity i and propagate them on line j in both directions, i.e., toward box $(i-1, j)$ and box $(i+1, j)$.

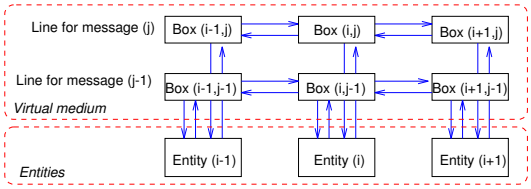

Figure 2. Bus type virtual medium.

- transmit j-type messages coming from box $(i-1, j)$ to box $(i+1, j)$ and vice versa, while duplicating them for entity i.

Figure 3 details box (i, j).

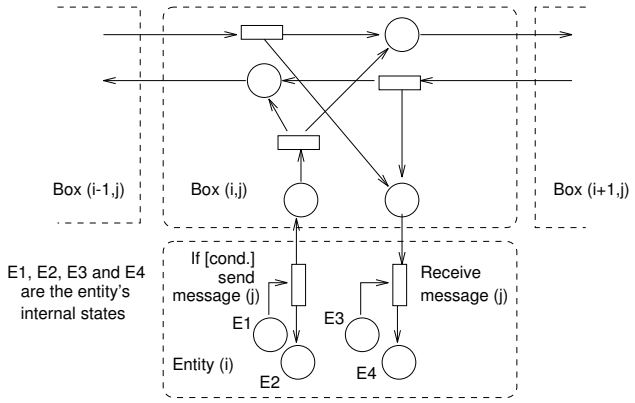

Figure 3. Inside of box (i, j).

Such a virtual medium liking n entities that is capable of exchanging m different sorts of messages, consists of m lines like those shown in figure 2. Hence, the medium's complexity does not depend on the number of communicating entities. The drawback with such a medium is that receivers of a message do not get it simultaneously. However, this is no problem when modeling asynchronous interactions.

As a summary, there are two techniques for modeling communication mediums that link n entities and support m different sorts of messages. One provides a loosely structured model whose complexity is $O(m \cdot n^2)$ links. The other provides a structured one whose complexity is $O(m)$ links (or $O(m \cdot n)$ boxes).

3.2 Petri Net Models for Assistant Agents

In a Petri sub-net that models a communicating entity [8], places represent internal states of those entities, transitions correspond to either the synchronizing of the reception of messages or actions agents perform, e.g., sending a message as a speech act.

In our framework [4], the agents' internal states and the actions the agents can accomplish are gathered in a state transition graph that describe their behavior. While Ferber's approach focuses on internal states we focus on behavioral states. See [9] for a detailed approach on the designing of interaction protocols in MAS. Figure 4 exemplifies a state transition graph for a simple negotiation protocol.

In this graph arrows go from the sending agent's state prior to sending a message and end on a state the addressed agent(s) reach after receiving the message. In its

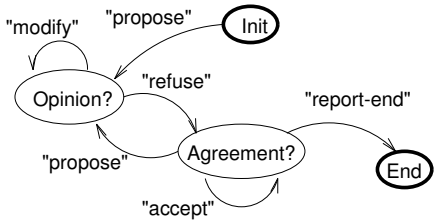

Figure 4. State transition diagram of a negotiation protocol.

initial ("Init") state an agent can send a proposal ("propose") which makes the receiving agent enter state "Opinion?". When an agent receives an modification (respectively acceptance) of proposal from another agent that is in state "Opinion" (respectively "Agreement?"), then it enters state "Opinion?" (respectively "Agreement?") too. In the "Opinion?" state an agent can also "refuse" a proposal which makes the receiving agent go to state "Agreement?". When the agents come to an agreement or want to abort the current negotiation they can stop negotiating ("report end") which makes the receiving agent enter state "End".

Let us see now the Petri sub-net model for agents whose behavior corresponds to the use of such a negotiation protocol.

Places are as follows:

- For internal states: `P:init`, `P:opinion?`, and `P:end`. We have merged states "Init" in state "Agreement?".
- For the receiving and storing of messages: `P:S-propose`, `P:S-accept`, `P:S-refuse`, `P:S-modify`, and `P:S-report-end`.

Transitions are:

- For the sending messages: `T:propose`, `T:accept`, `T:refuse`, `T:modify`, and `T:report-end`.
- For the reading of a received message: `T:R-propose`, `T:R-accept`, `T:R-refuse`, `T:R-modify`, and `T:R-report-end`.

An agent is in state s if the place that corresponds to this state holds a mark (token). This mark that is unique to each agent will be called a *state-mark*. It can only go to places denoting the agent's internal state.

The other kind of mark that is present denotes the messages sent. They are only located in places where messages can be received. Hence, in such a place there is as many marks as messages of that type received and not yet read by the agent.

At this point, places, transitions and marks are identified. A Petri sub-net modeling the agent can be built. Places and transitions need to be linked by an arc so that a change of internal state caused by various actions matches the description of an agent's behavior (figure 1). Figure 5 shows the corresponding Petri sub-net.

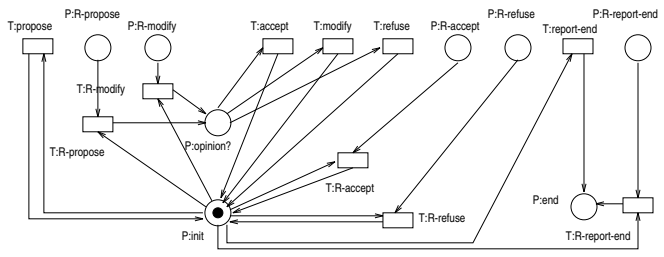

Figure 5. Petri sub-net modeling an agent that makes use of the negotiation protocol seen in figure 4.

There must be one token in the `P:init` agent's place—the state mark. This indicates the agent is in its initial state. From there, the agent is able to choose which action to perform and which transition to fire. This current Petri sub-net does not specify the way the transition to fire is chosen. As a matter of fact, interaction protocols do not provide this kind of information since this is the concern of the agent's decision. It falls within the application domain.

3.3 Complete Negotiation Protocol Model

A global Petri net can now be built in order to model our negotiation protocol. Petri sub-nets that correspond to agents have been defined. The virtual medium still needs to be chosen. Modeling the protocol and performing validation tests on a real application (with more than two agents) does not depend on the type of virtual medium. When there are very few communicating entities the loosely structured model whose complexity is $O(m \cdot n^2)$ links is bearable (see figure 1). On the other hand, for simulation reasons, when agents are complex and numerous there is a need for the simplest possible medium (i.e., bus type medium).

For space reason, no example of a final Petri net is shown here. But let us just mention that with a perfect virtual medium (the loosely structured one) places for getting and storing messages prior to their reading and storing by the agent, like `P:S-propose`, `P:S-modify`, etc. (see figure 5) belong in fact to the medium. Indeed, as shown in figure 1, the medium contains one place per agent and per type of message received. With a bus type medium, those places also belong to the medium and are duplicated in each of its boxes (see figure 2 and 3).

What is the theoretical significance of this complete negotiation protocol modeling? A protocol is a language L whose vocabulary V is the set of possible messages. V^* denotes the set of V's elements combinations, i.e., the set of all message sequences. The language L associated with the protocol is a sub-set of V^*. Given these definitions, we are building an automata A_1 by means of a Petri net, and whose language $L(A)$ is such that $L(A) \subseteq L$. In fact $L(A)$ is the restriction of L to a special case where n communicating entities are concerned. However, $L(A)$ can be viewed as a good approximation for L. The protocol validation can thus be carried out with the n communicating entities.

4 Protocol Pre-Validation

4.1 Theoretical Approach

Performing a protocol pre-validation boils down to conveying expected services and then check whether there is adequacy between those services and the ones the protocol really supplies. Needless to mention those services correspond to a verification of general properties that should characterize any protocol. Those properties have nothing to do with the application's semantic. We will call these *elementary services*.

Figure 6 explains the theoretical approach for verifying protocol properties. The first step is to go through a formal modeling of the protocol given in figure 4. This has been done in the previous section. Let us now convey the expected elementary services as far as our negotiation protocol. This amounts to check whether it satisfies general properties necessary to any protocol [5, 2].

- The first property one may wish is that the global protocol model, represented as a Petri net, be *bounded*. Indeed, a not bounded model means the protocol leads to a process with an infinite number of states.
- The second property is that the global protocol model be *quasi-livable*. If such a property is satisfied, all the model's transitions are fireable starting from the initial marking. This ensures all protocol's messages are used.

- Thirdly, it is important to know whether the protocol leads to deadlocks and if so what they are. Obviously in case of a negotiation protocol, all markings associated with a model's deadlock[4] (or *sink state*) must contain a token in each of the agents' "End" state.
- The last property one may wish to see satisfied is that the protocol's model must enable ending with the desired sink state situations. In other words, the negotiation process infered by the protocol must end.

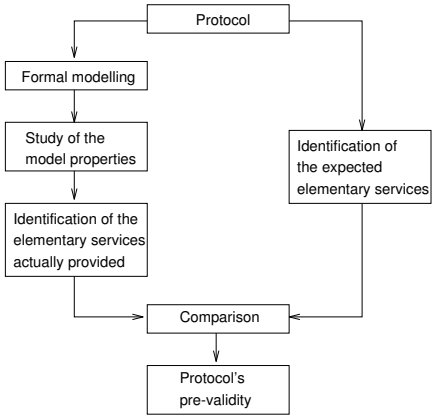

Figure 6. Principle of protocol pre-validation

This analysis being done let us express the expected elementary services:

(a) The negotiation process leads to a finite number of states.
(b) Every kind of protocol message is used.
(c) Every kind of sink state infered by the protocol corresponds to the end of the negotiation process.
(d) The negotiation process always reaches one of these sink states situations.

A number of other properties can be subjected to verification. However, these are specific to negotiation.

Having discussed the theoretical issues of a protocol's prevalidation principles, we will now detail more computational aspects

4.2 Use of a Simulation/Validation Tool

Editing the Petri nets that model negotiation protocols and analyzing certain general properties related to the net's structure have been carried out by means of a software tool [11]. Among other things, this tool allows for a graphical representation of Petri nets, their simulation, and the search for properties. In order to check on these properties, this software conducts an analysis of the net by means of its states; the state of a Petri net being given by its marking. At first, the software sets up the graph of reachable markings, i.e., the list of states the net can reach and the sequences of fireable transitions to go from one state to another.

The graph of markings may be infinite and therefore cannot be built. It is thus necessary to build the coverability graph which is some condensed version of the graph of markings. The purpose of such a graph is to obtain a sub-set of states that gives enough information on the set of reachable states, especially when it is infinite [3].

Subsequently, seeking properties consists in examining the coverability graph according to a procedure specific to each of the properties.

[4] A *deadlock* is a marking such that no transition is enabled.

4.3 Properties and Validity

Properties of the Petri net that model our negotiation protocol are given in the appendix of this article. In this section we discuss the properties found and infer the elementary services actually provided by the protocol.

1. The net is bounded. This implies that **(1) this Petri net leads to a finite number of states**. In other words, the negotiation process thus modeled has a finite number of states. Furthermore, there is no infinite sequence of transitions in this net. This does not help us identify the final states of these sequences. On the other hand, it implies that **(2) the negotiation process does not enter cyclic or infinite sequences but always reach a terminal state**.
2. In the final Petri net that model our negotiation protocol with three agents, we have made several hypotheses in order to restrict the number of reachable states (for a simulation purpose): agent 1 is the only one to send a proposal (agents 2 and 3 cannot "propose": `T:propose-2`, `T:propose-3`), agents 2 and 3 can only send modifications (agent 1 do not "modify": `T:modify-1`), and agent 1 is the only agent allowed to end the negotiation process (agents 2 and 3 do not "report end": `T:report-end-1`, `T:report-end-2`). Consequently, agent 1 does not receive and therefore read any "propose" message (`T:R-propose`). Indeed, messages are not sent to the sending agent even when in *broadcast* mode.
   Thus, the only non quasi-live transitions are those six transitions. In particular, to each forbidden transition in an agent there corresponds an equivalent transition in a neighbor agent that is not forbidden and quasi-live. Had we not precluded some transitions on purpose, all would have been quasi-live. Thus, **(3) the protocol leads to negotiation processes in which no message is unused**.
3. The expression of the terminal states with the coverability graph shows that the terminal markings have a common feature: none have a mark in place `P:init-x` and all have a mark in place `P:end-x`. In other words, the three agents reach their internal state "end" when the net is in a terminal state. Thus, **(4) all the net's terminal states correspond to the end of the negotiation process**.
4. Places `P:init-x`, `P:opinion?-x` et `P:end-x` are in mutual exclusion (i.e, it is impossible to have more than one mark in those places) and actually, there is always one mark in either of these places. Thus, **(5) at one time, agents (modeled in figure 5) have a unique internal state, and are never in a state different from "init", "opinion?" and "end"**.

All four expected elementary services (given in section 4.1) are achieved: conclusion (1) implies service (a) is achieved, conclusion (2) and (4) implies services (c) and (d) are achieved, and conclusion (3) implies service (b) is achieved.

5 Concluding Remarks and Future Work

In this article we described an approach that formalizes and partially validates interaction protocols for multiagents systems once protocols have been designed. This means that one needs first to identify the agents' behavior and define the state transition graph of the interaction protocol. Once this is done, we have seen

1. how to build a Petri sub-net model for the agents,
2. which virtual medium to chose in order to link the agents,
3. how to build a complete Petri net for modeling the entire protocol, and
4. what it takes to seek such a Petri net's properties, by means of a software tool capable of simulations.

Usually the issue of interaction in MAS is almost exclusively tackled from a Speech Acts theory perspective. But the approach we have taken in this paper stems rather from the

Distributed Systems' domain. Very few attempts have been investigated in this way (let us mention [13] though). We hope this paper will participate in offsetting this general tendency.

As far as future development, several domains need further work. One deals with the nature of properties to be verified. We have only touched on rather general properties that one wants to find in any interaction protocol. Those properties do not depend on the application. We are interested now in expressing and verifying semantic properties that are proper to the protocol given an application.

Appendix

Here is the file produced by PAPetry software for the seeking of properties in the Petri net modeling our interaction protocol.

```
+======================================+
          COVERING GRAPH
       P. FRAISSE and C.JOHNEN
+======================================+
 1 - Diagnosis
 2 - Unbounded places
 3 - Non quasi-live transitions
 4 - Home states
 5 - Live transitions
 6 - Termination
 7 - Fireable sequence from initial state
 8 - Reachable state
 9 - Expression of a state
10 - Mutual exclusion
11 - Print the net
12 - End of program
****************************************
Your choice: 1

The net admits 9000 states
It is bounded.
It does not admit infinite firing sequences.
****************************************
Your choice: 2
All the places are bounded.
****************************************
Your choice: 3

Non quasi-live transitions:
T:modify-1   T:report-end-1   T:R-propose-1
T:propose-2  T:report-end-2
T:propose-3  T:R-report-end-3
****************************************
No home state.
****************************************
Your choice: 5
No live transition.
****************************************
Your choice: 6
There are 1194 terminal components.

Component 1
 States 18

   (...)

Component 1194
 States 8977
****************************************
Your choice: 10
Enter groups of place names separated by a comma,
Enter a semicolon to end.

P:init-1,P:opinion?-1,P:end-1;

There is mutual exclusion and a group is always marked
****************************************
Your choice: 10
Enter groups of place names separated by a comma,
Enter a semicolon to end.

P:init-2,P:opinion?-2,P:end-2;

There is mutual exclusion and a group is always marked
****************************************
Your choice: 10
Enter groups of place names separated by a comma,
Enter a semicolon to end.

P:init-3,P:opinion?-3,P:end-3;

There is mutual exclusion and a group is always marked
+======================================+
            INITIAL MARKING
+======================================+
Place P:init-1, color black:       1
Place P:init-2, color black:       1
Place P:init-3, color black:       1
```

REFERENCES

[1] B. Burmeister, A. Haddadi, and K. Sundermeyer, 'Generic, configurable, cooperation protocols for multi-agent systems', in *From Reaction to Cognition*, eds., C. Castelfranchi and Jean-Pierre Muller, volume 957 of *Lecture notes in AI*, pp. 157–171, Berlin, Germany, (1995). Springer Verlag.

[2] J.-P. Courtiat, J.-M. Ayache, and B. Algayres, 'Petri nets are good for protocols', in *Symp. on Communications Architectures and Protocols*, Montréal, (June 1984). Sigcomm'84.

[3] René David and Hassane Alla, *Petri nets and Grafcet: tools for modelling discrete event systems*, Prentice Hall Int., Hertfordshire, UK, 1992.

[4] Yves Demazeau, 'From interactions to collective behaviour in agent-based systems', in *European Conference on Cognitive Science*, Saint-Malo, France, (April 1995).

[5] M. Diaz, 'Modelling and analysis of communication and cooperation protocols using petri net based models', in *2nd Int. Workshop on Protocol Specif., Testing and Verification*, ed., C. Sunshine, Idyllwild, CA, (May 1982). Proc. of the IFIP WG 6.1, North-Holland.

[6] Amal El-Fallah-Seghrouchni and Serge Haddad, 'A framework for agent coordination oriented distributed planning and resource sharing managment', in *7th MAAMAW workshop*, pp. 35–, Sweden, (13–16 May 1997).

[7] P. Estrailler and C. Girault, 'Applying petri net theory to the modelling analysis and prototyping of distributed systems', in *Proceedings of the IEEE Int. Workshop on Emerging Technologies and Factory Automation*, Cairns, Australia, (1992).

[8] Jacques Ferber, *Les systèmes multi-agents, vers une intelligence collective*, InterEditions, Paris, 1995.

[9] Jean-Luc Koning, Guillaume François, and Yves Demazeau, 'An approach for designing negotiation protocols in a multi-agent system', in *15th IFIP World Computer Congress, IT&KNOWS Conference*, Vienna (Austria), Budapest (Hungaria), (31 August–4 September 1998). Austrian Computer Society.

[10] Th. Kreifelts and F. von Martial, 'A negotiation framework for antonomous agents', in *Decentralized AI 2*, eds., Yves Demazeau and Jean-Pierre Muller. Elsevier Science Publishers B.V., (1991).

[11] PAPetri. Institut d'informatique et d'entreprise. ftp://ftp.cnam.fr/. public domain software.

[12] Sati Singh Sian, 'Adaptation based on cooperative learning in multi-agent systems', in *Decentralized AI 2*, eds., Yves Demazeau and Jean-Pierre Muller, pp. 257–272. Elsevier Science Publishers B.V., (1991).

[13] F. Vernadat, D. Azema, and A. Lanusse, 'Modélisation par réseau de petri d'un langage d'acteurs', in *2èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, Voiron, France, (9–11 May 1994). Afcet & Afia.