Synchronized Petri Net: A Formal Specification Model for Multi Agent Systems

Sofia Kouah
University of Oum El-Bouaghi, Algeria
MISC Laboratory, University Mentouri Constantine, Algeria
Email: kouah@misc-umc.org

Djamel Eddine Saïdouni
MISC Laboratory, University Mentouri Constantine, Algeria
Email: saidouni@misc-umc.org

Jean Michel Ilié
Lip6 Laboratory, University of Paris 6, Paris, France
Email: jean-michel.ilie@upmc.fr

Abstract—**This paper proposes a formal model for specifying multi agent systems, named SyPN (for Synchronized Petri Net). This model allows the specification of various kinds of agent-based systems' behaviors, such as individual and collective behaviors. SyPN is an extension of Recursive Petri net allowing synchronization of several nets. In fact, SyPN borrows the specification of dynamic processes from Recursive Petri net and introduces several valuable concepts that enable concise multi agent system specifications, such as: typed places, transitions and tokens, synchronization points, synchronization condition, synchronization relation and binding function. We illustrate our approach by two case studies of remote interactions between agents.**

Index Terms—**Multi-agent systems, formal specification, recursive petri nets, abstraction, synchronization.**

I. Introduction

Multi Agent Systems (MAS for short) form a powerful paradigm to design complex software [7][22][33]. Generally, it resolves complex problems where reactivity, mobility [30], dynamicity and adaptation of the system to uncertain or unpredictable factors should be considered. MAS may be seen as societies made up of autonomous and independent entities, called agents. These agents interact together in order to solve a specific problem or to achieve collectively a common task. Agent is viewed as a computer system situated in some environment and that is capable of executing flexible autonomous actions in this environment in order to meet its design objectives [35] [56].The large majority of applications based-agents are designed by [48]:

- Using methodologies based on results of Object-Oriented Software-Engineering [5][9][34];
- Highlighting organizational aspects [17] or relations between various aspects of MAS [10][40][44];
- Applying construction techniques of expert systems [19];
- Or using formal reasoning, based on Z- language [43] or on temporal logic [1].

Then, the domain for designing MAS is becoming very attractive. However, several functionalities of MAS like parallelism, dynamicity and communication may not be easily specified using existing specification models. In fact, these models are not defined specifically for MAS. Therefore, new specification models, new paradigms and new tools are required. Designing safe and sound MAS whose behaviors could be checked before its achievement, calls for a rigorous specification step, assisted by a formal specification model. This model should have a well-defined semantics and being able to take into account all MAS functionalities. Previous efforts toward modeling and verifying MAS can be found in the work of [29] [54]. They are based on reachability graph generation. In our work, this issue can be addressed likewise, by generating reachability graph from SyPN model. Such generation can be easily built by generating all possible firing sequences as follow:

- Step 1: start from initial marking.
- Step 2: fire all enabled transitions from the current marking (i. e. apply firing rules).
- Step 3: re-iterate the second step from the new marking until no transition can be enabled.

In this context, we are mainly interested by formal specification model of MAS. In this field some basic questions arise:

- How to preserve agent's proprieties, such as autonomy, sociability, awareness...etc.
- How to model dynamically agents' interactions.
- Asynchronous is a ubiquitous aspect characterizing concurrent interactions that must be straightforwardly modeled.

doi:10.4304/jsw.8.3.587-602

Several models were proposed in the literature, such as Petri nets [6], recursive Petri nets [15], colored Petri nets [16][45][57], finite state automata [52], Maude [46], Z language [50], Event-B [31][32]. As it will be shown in section 5, RPN has several advantages comparing to other models. Recursive Petri nets cover several functionalities of MAS such as modeling abstraction, dynamicity, concurrency, preemption, recursion …etc.

In fact, Recursive Petri net (RPN) supports the specification of concurrent activities, reasoning about simultaneous actions and continuous processes, a theory of verification and mechanisms of transformation (e.g. abstraction, refinement, merging) [3] [4] [13] [24] [27]. Furthermore, RPN has aptitude to model threads which behave concurrently [15][16][25][26]. They are able to create new threads, until their respective ends. RPN distinguish elementary and abstract transitions. Moreover a starting marking is associated to each abstract transition and a semi-linear set of final markings is defined. The firing of an elementary transition updates the current marking using ordinary firing rule. The firing of an abstract transition refines it by a new sub-net (i.e. creation of new thread, named its child) which starts its own token game, from a starting marking whose value is attached to the abstract transition. Once a final marking is reached, a cut step closes the corresponding sub net, kills its children and produces tokens, indicated by the post matrix, in the appropriate output places of the abstract transition.

However, there are some limitations [3]:

- In recursive Petri net, there is no composition between nets, then there is no way to model interactions between agents modeled in terms of RPNs.
- In RPN, processes are associated to the execution of an abstract transition; this execution is represented by the semantics associated to RPN. Then, the semantics doesn't allow any choice initiated by the process. We just generate the marking graph which gives all possible behaviors. This semantics limits RPN for allowing process autonomy.
- In RPN there is only one way to choose the refinement net. (i. e. which limits the dynamic refinement process of abstract transitions.)

To allow interaction between processes, in [8], authors extend RPN by shared places. Each process is specified by RPN and shared places ensure interactions between processes. This model is named ERPN (Enhanced RPN). It is enhanced by:

- The possibility, for a given process, to control creation of its processes (i.e. children) with a new kind of outputs associated to the abstract transitions, namely immediate outputs.
- New operational semantics that allows manipulation of both local and global places (shared places).

On the other hand, ERPN has many limits such as [8]:

- Some agent functionalities like autonomy and dynamic interaction may not be easily modeled in ERPN.
- It does not deal systematically with the following contrast: with which group communicate each instance of agent?

The main contribution of this paper is the proposition of a new formalism, named SyPN for (Synchronized Petri Net). As mentioned above, RPN doesn't allow the specification of agents' interactions. For this reason SyPN extends RPN by enabling dynamic interactions between several Petri nets, preserving agents' proprieties specially autonomy and using ERPN's shared place functionality. Effectively, interactions between agents are complicate and even uncontrollable [2]. When designing MAS, dynamic interaction should be well described and easily specified to ensure agents' adaptation to the dynamic environment changes. To adapt themselves to these changes, agents must be able to change their roles and behaviors under several circumstances and adopt various protocols to interact with agents.[55]

The paper is structured as follows: Section 2 presents MAS behaviors. Section 3 defines the SyPN model, presents a contextual definition of the model, its syntax, its graphical display and its formal semantics. Section 4 presents two examples that illustrates SyPN modeling facilities among others synchronization between tasks and remote interactions. Section 5 discusses some related work. Section 6 concludes this work and discusses prospects to be continued in future.

II. BEHAVIORAL CHARACTERISTICS OF MAS

A MAS is an organized collection of agents that can be characterized by all or some of the following features [17]:

- Each agent has incomplete information, or capabilities for solving the problem, thus each agent has a limited viewpoint;
- There is no global system control;
- Data is decentralized; and
- Computation is asynchronous.

Agent's behavior depends on its own characteristics, such as autonomy, reactivity, pro-activity (i.e. agents do not simply act in response to their environment. They are able to exhibit goal-directed behavior by taking an initiative [56]), sociability, goals oriented, dynamicity…etc. Each property is present in an agent with more interest than another, this depends on agent functionalities. Each agent has an identity, a state and a behavior. A state consists of agent's knowledge, beliefs and goals that it should perform. Agent's behavior is defined by roles it can perform (e.g. in E-Learning MAS [36], agent's role can be: *learner*, *instructor*, *author*, *reviewer*, *administrator*...etc.), actions it can achieve and its reactions to different events. A role can be thought of as a set of behaviors and capabilities that agents can exploit to perform their tasks in a given context [28].

Let us study agent's behavior, by making abstraction on its internal architecture or any implantation choice. In

MAS, an agent must be able to handle very large numbers of concurrent tasks or processes in order to react to various external events. To successfully ensure this management each agent achieves one or more behaviors.

As far as agent's behaviors are concerned, we adopt the following classification. Two main categories are distinguished: *simple behavior* being used to present simple tasks or processes and *complex behavior* being used to present complex tasks or processes.

Simple behavior: it includes cyclic, atomic and planed behavior.

- **Cyclic behavior**: this kind of behavior achieves its task recursively, for each call.
- **Atomic behavior**: it achieves a simple task and finishes. It includes three kinds of sub-behaviors:
  - *Sender behavior*: It encapsulates an atomic unit that achieves an operation of sending message.
  - *Receiver behavior*: This behavior waits until reception of a message. It achieves an atomic operation of receiving message then it stops.
  - *Generic behavior*: Models any behavior made up of any single atomic task other than sending or receiving messages.
- **Planed behavior**: This behavior achieves a task within certain time.

Complex behavior: it includes sequential, parallel, exclusive and combined behaviors:

- **Sequential behavior**: it models a sequence of actions.
- **Parallel behavior**: it models a collection of behaviors that must be executed concurrently.
- **Exclusive behavior**: it models the exclusive execution of several behaviors.
- **Combined behavior**: it models combination of various behaviors in order to produce more complex one.

III. SYNCHRONIZED PETRI NET SPECIFICATION MODEL

The main characteristic of the SyPN model is its ability to specify easily several functionalities related to remote agents such as communication and synchronization. Also, the model distinguishes local or internal behavior of an agent among collective agent's behaviors. Internal behavior is viewed as an organized collection of abstract actions which can be performed locally without interaction with other agents. In other words, an agent has its own requirements (i.e. capabilities and resources) to accomplish this kind of actions. Contrary to internal behavior, actions of collective behavior need to be coordinated with other agent actions. SyPN extends recursive Petri net by:

- Typed places, transitions and tokens;
- New concepts, which are: synchronization points, synchronization condition, synchronization relation and binding function.

A. SyPN Syntax

This section presents definitions and notations used for SyPN's formalization.

1) Typing of Place, Transition and Token:

Places Types: The set of places is partitioned into two subsets: local places and synchronization places which are noted respectively P_{Loc} and P_{Sy}. The local state of an agent is defined by its local places. The global state is composed of local and synchronization places.

Transitions types: Four types of transitions are distinguished:

- **Elementary transition**: It models a local elementary task. This transition does not need any refinement. The set of elementary transitions is denoted by T_{elem}.
- **Abstract transition**: It models a local abstract task. The execution of this task requires an adequate refinement that depends on execution context. The set of abstract transitions is denoted by T_{abs}.
- **Elementary synchronization transition**: It models an elementary task of synchronization which does not require any decomposition. The set of elementary synchronization transitions is denoted by T_{elemSy}.
- **Abstract synchronization transition**: Such transition is needed when an abstract task is associated to a synchronization occurrence between remote agents. The abstract task is defined by its refinement process. The concurrent execution of abstract synchronization transitions leads to concurrent execution of the corresponding abstract tasks. The set of the abstract synchronization transitions is denoted by T_{absy}.

Token types: Two token types are distinguished:

- **Local token**: It is used in firing elementary and abstract transitions.
- **Synchronization token**: It allows the evolution of several synchronization nets. This synchronization is defined by abstract synchronization transitions and elementary synchronization transitions.

Definition 1 (Token): A token is a pair (type, id) where $\text{type} \in \{\text{Local}, \text{Syn}\}$ represents the token's type and id represents the token's identifier. This identifier identifies a process (agent).

Notations:

- In the sequel, pairs $(Local, i)$ and (Syn, i) will be noted $Local_{\text{i}}$ and Syn_{i} respectively.
- Schematically, Fig. 1 presents notations that will be used.

Remark: As it will be clarified in the sequel, a synchronization transition (elementary and abstract) is characterized by a synchronization point allowing their distinction.

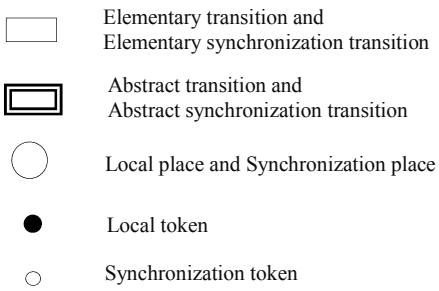


Figure1. SyPN annotations

Definition 2 (Net): A net R is a triplet (P, T, A) where:

- P is a finite set of places, such that $P = P_{Loc} \cup P_{Sy}$.
- T is a finite set of transitions, such that $T = T_{abs} \cup T_{elem} \cup T_{elemSy} \cup T_{absSy}$;
- A is a finite set of edges linking places and transitions such that $A = A_E \cup A_S$, where $A_E \subseteq P \times T$, $A_S \subseteq T \times P$.

Definition 3 (Marked Net): A marked net Rm is a pair (R, M) where:

- $R = (P, T, A)$ is a net;
- $M : P \to \mathbb{N} \times \mathbb{N}$, is a marking function such that: $\forall p \in P, M(p) = (x, y)$, where x is the number of local tokens in the place p and y is the number of synchronization tokens in the place p. The number of local tokens and synchronization tokens are be also noted $M(p).loc$ and $M(p).syn$ respectively. In other words $M(p).loc = x$ and $M(p).syn = y$.

Example 1: Fig. 2 represents a marked net, having three local places P_1, P_2 and P_3, three transitions T_1, T_2 and T_3, such that $T_1 \in T_{elem}$, $T_2 \in T_{abs}$ and $T_3 \in T_{absy}$ and the following edges: $(P_1,T_1),(P_1, T_3),(T_1, P_2),(T_3, P_3),(P_2, T_2)$ and (T_2, P_3). The marking of these places is defined by:

$$M(P_1) = (1,1);\ M(P_2) = (0,0);\ M(P_3) = (0,0).$$

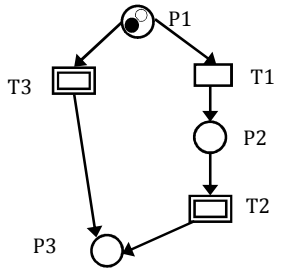


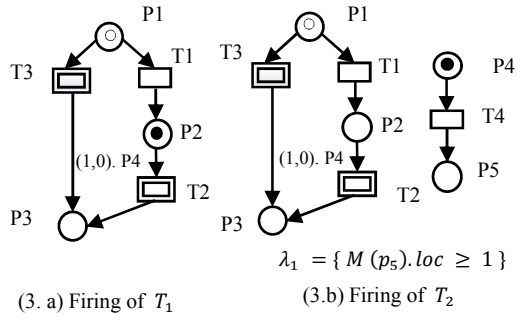
Figure 2. Example of a marked net.

B. The intuition and Definitions behind SyPN

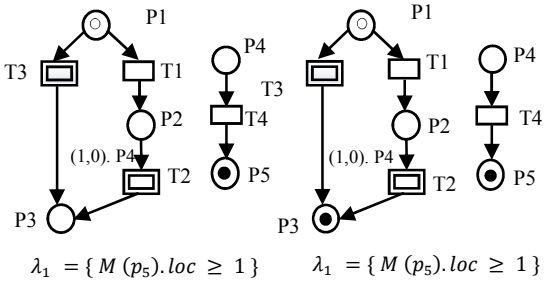
Elementary and abstract transitions: The firing of elementary transitions is similar to the firing of transitions in Petri nets. However, the firing of abstract transitions is similar to the firing of abstract transitions in recursive Petri nets, where two main phases are distinguished: consuming tokens defined by pre-condition of the abstract transition and creating a new process. This process is modeled by a net, called the refinement net.

Refinement net has an initial marking and a semi-linear set of final markings. One or more refinement nets are associated to each abstract transition. The choice of refinement net is done by a binding function (see Definition 6). It should be noted that firing conditions of elementary and abstract transitions depend on local tokens only. The intuition of firing abstract transition is analogous to both RPN and SyPN models. However, RPN doesn't define how the appropriate refinement net is chosen. As noted previously, SyPN uses a binding function to dynamically find a suitable refinement net (i.e. non deterministic).

To clarify the idea, consider the SyPN of Fig.2. From this state, the elementary transition T_1 is enabled. The marking which is resulted from the firing of T_1 is shown in Fig.3.a. From this state, the transition T_2 is enabled. The firing of T_2 is achieved in two steps: The consumption of local tokens specified by the pre-condition of T_2 (i. e. token of place P_2) and the creation of a process, modeled by a refinement net (see the net at net on the right of Fig.3.b). This net starts its evolution with an initial marking (represented, in Fig.3.c by the annotation $(1,0).P_4$ associated to the transition T_2). A set of final marking is defined in order to describe its termination (i.e. $\lambda_1 = \{M(p_5).loc \geq 1\}$). Starting from the configuration described by Fig.3.b, the transition T_4 is also enabled. The firing of transition T_4 produces a token in place P_5 (i. e. Fig. 3.c).



(3. a) Firing of T_1 (3.b) Firing of T_2



(3.c) Firing of T_4 (3.d) Firing of the cut τ_1

Figure 3.Evolution of the net of Fig. 2.

In fact, the main interests of refining abstract transitions are:

- Allowing concise modeling and easier verification. In other words, refinement process offers the designers a scalable support for the development of

MAS. In a top-down refinement-based approach, development starts from an abstract specification level of a system that models the most necessary functional requirements in non-deterministic manner. A sequence of refinement steps is needed to gradually reduce non-determinism and introduce more details [49].

- Allowing flexible capturing of agents properties such as autonomy, awareness, dynamicity, etc.

Local cut-steps: Several cut-steps for refinement net may be distinguished. Each local cut-step τ_i corresponds to final marking set λ_i. Once a final marking is reached the corresponding cut-step will take place. At first, it destroys process which is created by the firing of the abstract transition that gave birth to this process. Then, it produces tokens indicated by the post-conditions of this transition.

For instance, the marking that results from the firing of transition T_4 (Fig.3.c), corresponds to a configuration in which the process can be finished by achieving a local cut-step. This cut-step is restricted by the final marking indicated by the set λ_1. The firing of the cut-step τ_1 is done in two steps: destroying the created process from the firing of the abstract transition T_2 and producing a token in the output places of the transition T_2 (Fig.3.d).

Synchronization condition: Synchronization condition is a relevant aspect to model synchronization between several nets in SyPN model. The main interest of this concept is for capturing autonomy agent's property and expressing clearly conditions that must be fulfilled to join a rendez-vous. Agents participate to a rendez-vous if they intend to such interaction and the necessary conditions of this interaction fulfilled. The first condition captures autonomy property of agents however the second one expresses the constraints to be satisfied. This condition concerns only incoming edge of abstract synchronization transitions. As an example, see the synchronization condition (x *and* y) of figure 5.

Elementary synchronization transitions and abstract synchronization transition: Contrary to the refinement of abstract transitions, the refinement of abstract synchronization transitions is constrained by a condition. This condition is called synchronization condition (see Definition 4). It also invokes, simultaneously with other abstract synchronization transitions, a refinement net which differs from the refinement nets associated to abstract transitions. This net is named synchronization point. The synchronization points model all synchronization's relations between nets (Definition 5). A synchronization point is defined by a marked net which is described by:

- Synchronization condition (pre-conditions),
- Synchronization relation,
- Synchronization places,
- Synchronization elementary transitions,
- Synchronization tokens,
- Initial marking and semi-linear sets of final markings.

Example 2: Fig. 4 shows the graphical representation of a synchronization point having three synchronization places (P_{11}, P_{12}, P_{13}) and two elementary synchronization transitions (T_{11}, T_{12}).

Remarks and notations:

- Nets associated to synchronization points depend on the synchronization relation.
- PS denotes the set of nets related to synchronization points. The set of other nets is noted R with the hypothesis that PS and R are disjoint.

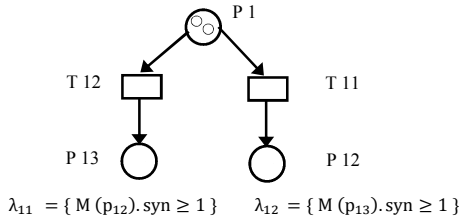


Figure 4. Synchronization point.

Synchronization relation: Each abstract synchronization transition is labeled by a synchronization relation. The intuition behind this labeling is to specify, for n nets and n abstract synchronization transitions the synchronization relation (i.e. the rendez-vous) to be ensured by a synchronization point. Thus, a synchronization relation is a relation between abstract synchronization transitions of n nets according to a synchronization operator.

As an example, the synchronization relation ($R_1.T_{11} \parallel R_2.T_{12}$) in figure 5 relates transitions T_{11} and T_{12} of respectively nets R_1 and R_2; according to the parallel synchronization operator.

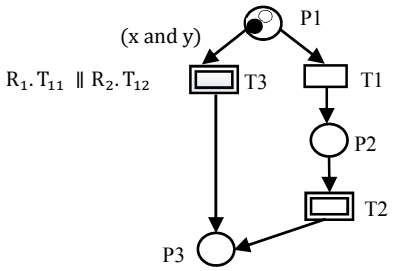


Figure 5.Synchronization condition and synchronization relation

A synchronization operator Op is defined by the following BNF syntax:

$Op ::= + \mid \parallel \mid \parallel\parallel\parallel \mid >> \mid \oplus$. Where:

- + is the nondeterministic choice.
- $\parallel$ is the synchronous parallel composition.
- $\parallel\parallel\parallel$ is the asynchronous parallel composition.
- >> is the sequential composition.
- $\oplus$ is the exclusive composition.

Note that, the operators "$\parallel$, $\parallel\parallel\parallel$" have different meaning: "$\parallel$" is the synchronous parallel composition. This operator relates parallel abstract synchronization transitions which must be synchronized (rendez-vous). However, "$\parallel\parallel\parallel$" is the asynchronous parallel composition. Parallel abstract synchronization transitions evolve asynchronously (i.e. independently).

Formally, synchronization condition and synchronization relation are defined as follows:

Definition 4 (Synchronization condition): Let I be a finite set of indices; $\mathrm{var} = \{x_i \text{ such that } i \in I\}$ is a finite set of variables.

The set of synchronization condition ranged over by $\beta_1, \beta_2, \ldots, \beta_i \ldots$ is defined inductively as follow:

- $\mathrm{Con}(x)$, a boolean expression of parameter the variable x . $\mathrm{Con}(x)$ is an elementary synchronization condition. As an example, for a variable x of natural numbers, x > 5 is an elementary condition.
- For synchronization conditions β_1 and $\beta_2, \mathrm{not}(\beta_1), (\beta_1 \mathrm{or} \beta_2)$ and $(\beta_1 \mathrm{and} \beta_2)$ are synchronization conditions.

Definition 5 (Synchronization relation):

- Let I be a finite set of indices ranged over by, $i, j, \ldots$ and let R be a finite set of nets ranged over by $R_i, R_j, \ldots$
- Let Messages be a set of messages ranged over by Msg, that can be sent and received by processes (i.e. agents). We assume that elements of "Messages" respect ACL-Language[1] (Agent Communication Language) syntax, such as KQML [18] and FIPA-ACL [20] [21].
- The set of senders is ranged over by Sender.
- The set of receivers is ranged over by Receiver.
- Let GetReceiver ( ) be a function that may be invoked by a sender process. This function returns as a result the set of potential receivers of the sent messages.

According to these definitions and notations, the synchronization relation Rsy built on the set of nets R and a set of abstract synchronization transition T (ranged over by T_k, T_l) is defined by:

$\mathrm{Rsy} ::= \emptyset \mid R_i.T_k \ \mathrm{Op}\ R_j.T_l$

$\mid \mathrm{Send}(\mathrm{Sender}, \mathrm{GetReceiver}(), \mathrm{Msg})$

$\mid \mathrm{Receive}(\mathrm{Receiver}, \mathrm{Sender}, \mathrm{Msg})$[2].

with $i \neq j$ and Op a synchronization operator such that $T_k \in R_i$ and $T_l \in R_j$.

GetReceiver ( ) must be evaluated during the sending operation.

Where: $\mathrm{Send}(\mathrm{Sender}, \mathrm{Rset}, \mathrm{Msg})$ means that a sender agent send a message Msg to receivers of the set Rset (the result of GetReceiver( ) function) and $\mathrm{Receive}(\mathrm{Receiver}, \mathrm{Sender}, \mathrm{Msg})$ means that Receiver receives a message from a process (i.e. agent).

Sending a message is an asynchronous operation so the sender will not wait for the message either to arrive at the destination or to be received.

(1): The reader is invited to see [18] and [21] for messages examples written in ACL-Language.

(2): in Send(Sender,Rset,Msg) and Receive (Receiver,Sender,Msg) operations, the reader may note that Sender and Receiver parameters may be known from the context. However we leave them according to ACL-Language syntax.

Notations:

- Let e be an incoming edge of the transition t, $\mathcal{FB}(e)$ denotes a function that evaluates the synchronization condition associated to e. Which constrains the firing of transition t.
- $\mathcal{FRsy}(t)$ denotes a function that returns the synchronization relation. This synchronization relation is the label of the abstract transition t.
- We assume that:
  - For any incoming edge e of an abstract transition, $\mathcal{FB}(e) = \mathrm{false}$.
  - For any abstract transition t in T_{abs} , $\mathcal{FRsy}(t) = \emptyset$.

Let us now consider the firing of abstract synchronization transitions providing from different nets and contributing to the same rendez-vous. The necessary and sufficient condition of such firing is that all synchronization conditions corresponding to these transitions are fulfilled. To illustrate this refinement process, let us consider the example of Fig. 7. In the initial configuration of this system, it is clear that the necessary and sufficient condition for the simultaneous firing of T_3 and T'_1 is satisfied (i.e. $(x \text{ and } y) = \mathrm{true}$ and $(r \text{ and } s) = \mathrm{true}$). This firing creates a new process having as control structure a synchronization point (shown in Fig. 4). The created process starts its execution from its initial marking (i.e. $M(P_{11}) = (0,2)$). From the current marking concerning the processes in which the abstract synchronization transitions are enabled, the firing of these transitions consists of consuming tokens indicated by the pre-conditions of the abstract synchronization transitions (here a token Syn_{R_1} of the place P_1 and a token Syn_{R_2} of place P'_1).

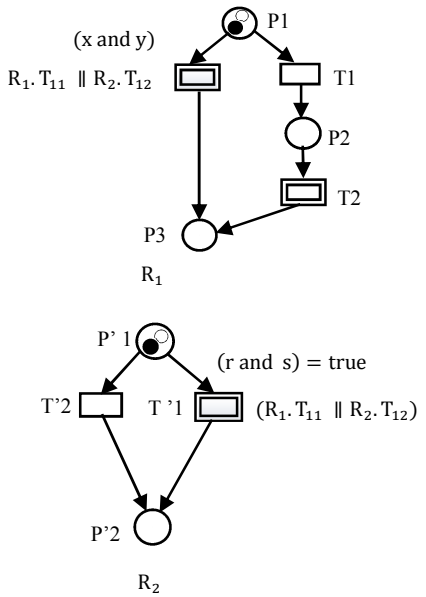
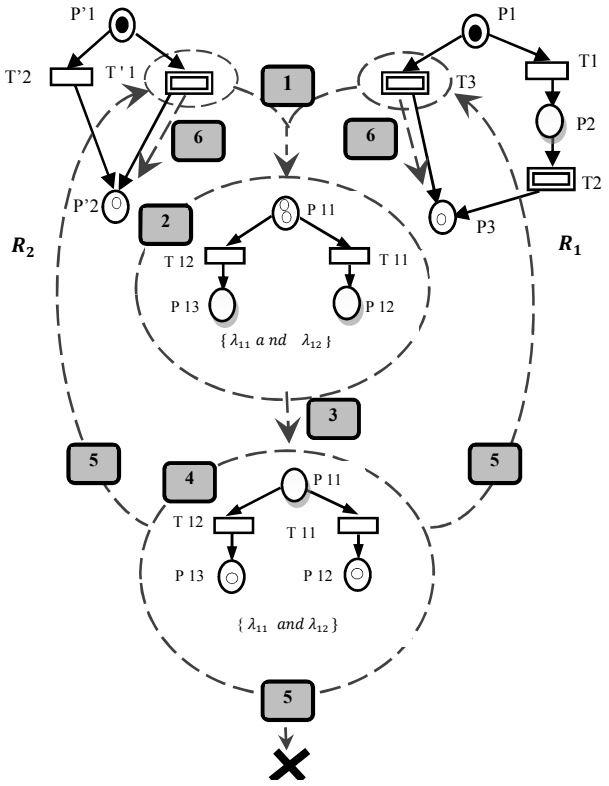


Figure 6. Firing of the abstract synchronization transitions.

Synchronization points and synchronization cut-steps: Let us clarify the intuition of the synchronization point through the example shown in Fig. 4: In this initial configuration, transitions T_{11} and T_{12} are enabled. T_{11} is enabled by Syn_{R_1}, however T_{12} is enabled by Syn_{R_2}. The

firing of T_{11} consumes the token Syn_{R_1} of the place P_{11}, however the firing of T_{12} consumes the token Syn_{R_2} of the place P_{11}. Marking that results from these firing consists of producing a token Syn_{R_1} in the place P_{12} and a token Syn_{R_2} in the place P_{13}. The configuration described; by marking that results from the firing of T_{11} and T_{12} corresponds to a configuration in which the two processes can be finished by carrying out a synchronization cut-step (see Fig. 7).



Caption:
1: Concurrent firing of T_3 and T'_1.
2: Creation of a synchronization point.
3: Concurrent firing of T_{11} and T_{12}
4: Production of tokens in the outgoing places of T_{11} and T_{12}.
5: Common termination of threads associated with T_3 , T'_1 and destruction of the synchronization point.
6: Production of tokens in the post-condition of T_3 , T'_1.
×: Destruction

Figure 7: Refinement process of abstract synchronization transitions and a possible cut-step.

Two cases are distinguished for this type of cut-step:

- **Differed termination**: The processes implied in the synchronization point have different termination's configurations. To express this kind of termination, we introduce the operator "and" between the sets of final markings associated to these processes. Thus, each terminated process will be destroyed. Once the last process is terminated and destroyed, the synchronization point will be achieved.
- **Common termination**: The processes implied in the synchronization point converge to the same termination. The final markings associated to these processes contribute to the formalization of this termination, expressed by a conjunctive combination of the final's markings sets. Thus, the synchronization point is achieved as soon as all processes terminate their executions. Terminated processes are destroyed simultaneously.

Example 3: Fig. 7 presents succession of firing for a refinement process that concerns the abstract synchronization transitions (T_3, T'_1) and a common cut-step associated to the termination set $\{\lambda_{11} = \{M(p_{12}).loc \geq 1\}$ and $\lambda_{12} = \{M(p_{13}).loc \geq 1\}\}$. See the caption of figure 7 for more explanation.

C. SyPN Formalization

Definition 6 (Synchronized Petri Net): Let SP be a synchronized Petri net. SP is a sextuple $(Rm, W, Var, Binding, \Omega, \Upsilon)$ such that:

- $Rm = (R, M)$ where $R = (P, T, A)$ is a marked net.
- W is an incidence matrix defined as follows:
  - $W^-(p,t) =$
    $\begin{cases} W_1^-(p,t) \text{ if } t \in T_{abs} \cup T_{elem}. \\ W_2^-(p,t) \text{ if } t \in T_{absSy} \cup T_{elemSy}. \end{cases}$
    Where: $W_1^- : P_{Loc} \times (T_{abs} \cup T_{elem}) \to \mathbb{N} \times \mathbb{N}$, and
    $W_2^- : P \times (T_{absSy} \cup T_{elemSy}) \to \mathbb{N} \times \mathbb{N}$.
  - $W^+(p,t) =$
    $\begin{cases} W_1^+(p,t) \text{ if } t \in T_{abs} \cup T_{elem}. \\ W_2^+(p,t) \text{ if } t \in T_{absSy} \cup T_{elemSy}. \end{cases}$
    Where $W_1^+ : P_{Loc} \times (T_{abs} \cup T_{elem}) \to \mathbb{N} \times \mathbb{N}$ and
    $W_2^+ : P \times (T_{absSy} \cup T_{elemSy}) \to \mathbb{N} \times \mathbb{N}$.
- Var is a set of net's variables.
- Binding is a function defined as follows: for e an incoming edge of a transition t, $Binding(t, \mathcal{FB}(e), \mathcal{FRS}y(t)) =$

$$\begin{cases} r \text{ where } r \in R. \text{ if } ((t \in T_{abs}) \wedge \mathcal{FB}(e) = \text{false} \\ \qquad \wedge \mathcal{FRS}y(t) = \emptyset). \\ ps \text{ where } ps \in PS. \text{ if } ((t \in T_{absSy}) \wedge \mathcal{FB}(e) = \text{true} \\ \qquad \wedge \mathcal{FRS}y(t) \neq \emptyset). \\ indfined\ otherwise. \end{cases}$$

It associates to each abstract transition a refinement net, and to each abstract synchronization transition a synchronization point according to the type of this synchronization (i.e. relation of synchronization).

- Initial marking function, defined by:
  $\Omega(t) = \begin{cases} \Omega_1(t) \text{ if } t \in T_{abs}. \\ \Omega_2(t) \text{ if } t \in T_{absSy}. \end{cases}$, Where:
  - $\Omega_1(t)$: A function associating an initial marking to the refinement net related to each abstract transition.
  - $\Omega_2(t)$: A function associating an initial marking to the synchronization point related to each abstract synchronization transition.
- Υ : A semi-linear set of final markings, defined by $\Upsilon = \Upsilon_1 \cup \Upsilon_2$ where :

— Υ_1: Indexed family of final markings, associated to the refinement's net concerning abstract transitions.
— Υ_2: Indexed family of final markings associated to the synchronization points.

Definition 7 (Extended marking): An extended marking of a synchronized Petri net noted $SP = (Rm, W, Var, Binding, \Omega, \Upsilon)$ is a labeled tree $Tr = (S, M, E, A)$, where:

- S is the finite set of nodes, where each node $s \in S$ is labeled by a marked SyPN , noted $< SP(s), M(s) >$.
- M is a marking function from $S \times P$ to $N \times N$.
- $E \subseteq S \times S$, is the set of edges, such that $E = \{< s, s' > s' \text{ is the child of } s\}$;
- A is a labeling function from $E \to (R \times T_{abs} \times Local) \cup 2^{(PS \times T_{absy} \times Syn)}$ such that Syn explores the set of synchronization tokens.

Extended marking is a labeled transition system.

A marked synchronized Petri net (SP, Tr_0) is a synchronized Petri net SP together with an initial extended marking Tr_0.

Notations[1]

For each node $s \in S$, $Succ(s)$ denotes the set of its direct and indirect successors including s ($\forall s \in S$, $Succ(s) = \{s' \in S \mid (s, s') \in E^*\}$ the transitive closure of child relation).

- Moreover, when s is not the root of the tree; pred (s) denotes its single predecessor.
- The empty tree noted $\perp$.
- When we treat several nets, the elements of the net are noted by a function (example: local places of a network r are noted by $P_{Loc}(r)$).
- Thereafter, we admit that for each synchronization cut-step τs ; a conjunctive or disjunctive combination of termination's sets is associated. These sets are indexed by λ_{ij}, where i identifies the net contributing to the synchronization point and j identifies the transition having invoked this synchronization. The intuition under this indexing is the preservation of processes identifiers having invoking this synchronization point.

D. Formal Semantics of Synchronized Petri Net

The operational semantics of synchronized Petri net is given in terms of state and change of state. If the current state of Petri net is completely defined by its marking, the state of a synchronized Petri net is defined as a tree of marked synchronized Petri nets, built by following successive refinements of both abstract transitions and abstract synchronization transitions. The intuitive interpretation of a state is as follows:

- The root of the tree corresponds to the initial state; it includes all synchronized Petri nets of the system.
  Each edge represents either a firing of abstract transition or a concurrent firing of the abstract synchronization transitions concerned by the same synchronization relation; where the extremity of the edge designates an unfolded net (i. e. synchronized Petri net) or a synchronization point.
- Each node is a set of marked synchronized Petri nets.

This tree evolves according to the synchronized Petri net's semantics, given through the firing rules of each type of transition. These rules will be detailed thereafter.

Definition 8 (Firing of elementary transition and elementary synchronization transition): An elementary transition (respectively *elementary synchronization transition)* t is enabled for a synchronized Petri net SP_i, a local token (respectively synchronization token) $token_i$, from a node s, of an extended marking $Tr = (S, M, E, A)$, noted $Tr \xrightarrow{SP_i, t, token_i, s} Tr' = (S', M', E', A')$ if and only if for any p in P_{loc} (respectively in P_{Sy}) , $M(s)(p) \geq W^-(p, t)$. Such that:

- $S' = S$;
- $\forall s' \in S \setminus \{s\}, M'(s') = M(s')$.
- For any p in P_{Loc}(respectively in P_{Sy}) , $M'(s)(p) = M(s)(p) - W^-(p, t) + W^+(p, t)$.
- $E' = E$.
- $\forall e \in E, A'(e) = A(e)$.

Definition 9 (Firing of an abstract transition): An abstract transition t is enabled for a synchronized Petri net SP_i, a token $local_i$, from a node s of an extended marking $Tr = (S, M, E, A)$, noted by $Tr \xrightarrow{SP_i, t, local_i, s} Tr' = (S', M', E', A')$, if and only if $\forall p \in P_I, M(s)(p) \geq W^-(p, t)$ such that:

- Let s' be a fresh identifier, such that: $s' \notin S$ and $Binding(t, \beta(e), Rsy(t)) = s$, with e is an incoming edge of t.
- $S' = S \cup \{s'\}$.
- $\forall s'' \in S \setminus \{s\}, M'(s'') = M(s'')$.
- $\forall p \in P_{Loc}, M'(s)(p) = M(s)(p) - W^-(p, t)$.
- $M'(s') = \Omega_1(t)$ is the initial marking associated to the refinement net.
- $E' = E \cup \{(s, s')\}$.
- $\forall e \in E, A'(e) = A(e)$.
- $A'((s, s')) = (SP_i, t, local_i)$
- $\Upsilon(t) = \Upsilon_1(t)$.

Definition 10 (Firing of abstract synchronization transition): Let t_i be an abstract synchronization transition, for a synchronized Petri net SP_i, a synchronization token Syn_i from a node s.

- Let $|t_i| = \{t_i\} \cup \{t \text{ where } \mathcal{FRsy}(t) = \mathcal{FRsy}(t_i)\}$.
- Let $R^i = \{ Sp \text{ where } t \in (|t_i| \cap T_{absy}(Sp))\}$.
- Let $|Syn| = \{ Syn_j \text{ where } t_j \text{ is enabled by } Syn_j \}$.

The simultaneous firing of the transitions t ($t \in |t_i|$), for the synchronized Petri nets designated by R^i, the tokens designated by |Syn|, from a node s of an extended marking $Tr = (S, M, E, A)$, denoted by $Tr \xrightarrow{R^i, |t_i|, |Syn|, s} Tr' = (S', M', E', A')$ is possible if and only if:

(1) Some notations are borrowed from RPN model [26].

$(((\forall\ Sp \in R^i, t \in (|t_i| \cap T_{absy}(SP_i)), (\forall p \in P_{Loc}(SP_i),\ M(s)(p) \geq W^-(p,\ t)) \&\ (\forall t \in |t_i|,\ \exists e / e$ is an incoming edge of $t,\ \mathcal{FB}(e) = true)))$ Such that:

- Let s' be a fresh identifier, such that $s' \notin S$ and $Binding(t, \mathcal{FB}(e), \mathcal{FRSY}(t)) = s'$.
- $S' = S \cup \{s'\}$;
- $\forall s'' \in S \setminus \{s\}, M'(s'') = M(s'')$.
- $\forall SP_i \in R^i,\ t \in (|t_i| \cap T_{absy}(SP_i)),\ \forall p \in P_{Loc}(SP_i), M'(s)(p) = M(s)(p) - W^-(p,t)$.
- $M'(s') = \Omega_2(t)$, initial marking associated to the synchronization point.
- $E' = E \cup \{(s, s')\}$.
- $\forall e \in E, A'(e) = A(e)$.
- $A'((s,s')) = 2^{(R^i \times |t_i| \times |Syn|)}$.
- $\Upsilon(t) = \Upsilon_2(t)$.

Definition 11 (Firing of local cut-step): The firing of a local cut step τ_i relating to the set of final marking γ_i from a node s for an extended marking $Tr = (S, M, E, A)$ leads to the extended marking $Tr' = (V', M', E', A')$, noted $Tr \xrightarrow{\tau_i, s} Tr'$, if and only if $M(s) \in \Upsilon_i$ such that :

- If s is the root of the tree and one of final markings is reached, the reduction leads to the empty tree (i.e. $Tr' = \perp$).
- Else, the semantics of the cu-step is as follow:
  - $S' = S \setminus succ(s)$.
  - $\forall s' \in S' \setminus \{pred(s)\}, M'(s') = M(s')$.
  - $\forall p \in P_{Loc}, M'(pred(s))(p) = M(pred(s))(p) + W^+(p,t)$.
  - $E' = E \cap (S' \times S')$.
  - $\forall e \in E, A'(e) = A(e)$.

Definition 12 (Firing of synchronization cut-step): A synchronization cut-step τs_i relating to the set of final markings $\Upsilon_2(t)$, such that t belongs to the set of transitions having contributed to the creation of a node s, is enabled from that node, for an extended marking $Tr = (S, M, E, A)$, noted $Tr \xrightarrow{\tau s_i} Tr' = (S', M', E', A')$, if and only if the final marking described by $\Upsilon_2(t)$ is reached. Two cases may be distinguished:

Differed Cut-stept:

- If $\Upsilon_2(t)$ has a disjunctive combination form of termination's set associated to the synchronization point then:
  - If s is the root of the tree and one of final markings is reached, the reduction leads, after exploring all final markings, to an empty tree (i.e. $Tr' = \perp$).
- Else, the semantic of synchronization cut-step relating to the set of final markings $\Upsilon_2(t)$ is as follows:
  - Let $\lambda = \{\lambda_{ij} / \lambda_{ij}$ Appear in $\Upsilon_2(t)\}$.
  - If ($\lambda_{ij} \in \lambda$) and (λ_{ij} is reached) then:
    - $\forall p \in P(SP_i), M'(pred(s))(p) = M(pred(s))(p) + W^+(p, t_j)$.
    - $\lambda = \lambda - \{\lambda_{ij}\}$.
  - If ($\lambda \neq \emptyset$) then:
    - $S' = S \setminus succ(s)$.
    - $\forall s' \in S' \setminus \{pred(s)\}, M'(s') = M(s')$.
    - $E' = E \cap (S' \times S')$.
    - $\forall e \in E, A'(e) = A(e)$.

Common Cu-stept:

- If $\Upsilon_2(t)$ has a conjunctive combination form of termination's sets associated to the synchronization point then:
  - If s is the root of the tree and all final markings are reached, then the reduction leads to an empty tree ($Tr' = \perp$).
- Else, the semantic of synchronization cut-step relating to the set of final markings $\Upsilon_2(t)$ is as follows:
  - Let $\lambda = \{\lambda_{ij}$ where λ_{ij} appear in $\Upsilon_2(t)\}$.
  - $\forall \lambda_{ij} \in \lambda$, λ_{ij} is reached then:
    - $S' = S \setminus succ(s)$.
    - $\forall s' \in S' \setminus \{pred(s)\}, M'(s') = M(s')$.
    - $\forall SP_i \in R^i, t \in |t_i|$, $\forall p \in P(SP_i), M'(pred(s))(p) = M(pred(s))(p) + W^+(p,t)$.
    - $E' = E \cap (S' \times S')$.
    - $\forall e \in E, A'(e) = A(e)$.

IV. Synchronized Petri Net Expressivity

This section gives two examples showing the SyPN modeling adequacy for MAS.

A. Generic Example of MAS Planning (synchronization between agents' tasks)

In fact, Multi-agent Planning (MAP for short) [12] [23] [37], extends traditional Artificial Intelligence (AI) planning to domains where multiple agents are involved in a plan and need to act together. Research in MAP is promising for real-world problems, as well to provide powerful tools for solving problems and ensure coordination between agents [13] [14] [38]. In fact, RPN has been proposed especially to model MAP problem [15].

Let us consider three agents A, B, and C which collaborate together in order to achieve a common goal concretized by a multi agent plan, noted Plan. Agents' capabilities and tasks' constraints are described as follow:

- Capability (A) = $\{T_1, T_6, T_8\}$,
- Capability (B) = $\{T_2, T_4\}$
- Capability (C) = $\{T_3, T_5, T_7\}$.
- T_1 and T_2 could be achieved independently.
- The post condition of T_1 enables the precondition of both T_5 and T_6.
- The post condition of T_4 enables the precondition of T_3.
- The executions of both T_5 and T_6 are in mutual exclusion.
- T_8 and T_9 could be performed concurrently.

Each task T_i has a pre-condition and a post-condition, respectively denoted $PreCond.T_i$ and $PostCond.T_i$. A pre-condition represents the necessary conditions that must be fulfilled before the task execution and, a post-condition represents necessary condition that must be

fulfilled after the task execution. . Tasks plan should be performed within the following sub order, by respecting tasks' constraints:

Plan $= (T_1 || |T_2,\ T_4 << T_3, T_5 \oplus T_6, T_7 | |T_8\}$.

Observe that at this level, any agents' internal behavior is voluntarily abstracted. Fig. 8 illustrates a possible modeling of example 1 using the SyPN formalism. Initially, the agent B can simultaneously interact with both agents A and C by invoking respectively the synchronization points SP2 and SP1. SP1 models the part $(T_1 || |T_2)$ and SP2 models the part $(T_4 << T_3)$, such that both parts can be executed simultaneously.

- SP 1 can be invoked if and only if TA1 and TB1 are simultaneously fired. This implied that TA1 and TB1 must be enabled (i.e.Gard1 is satisfied and $(M(PA1).syn \geq 1\ and\ M(PB1) \geq 1)$).
- SP 2 can be invoked if and only if TB2 and TC1 are simultaneously fired. This implied that TB2 and TC1 must be enabled (i.e. Gard2 is satisfied and $(M(PB1).syn \geq 1\ and\ M(PC1) \geq 1)$).
- SP 3 can be invoked if and only if TA2 and TC2 are simultaneously fired. This implied that TA2 and TC2 must be enabled (i.e. Gard3 is satisfied and $(M(PA2).syn \geq 1\ and\ M(PC21) \geq 1)$).
- SP 4 can be invoked if and only if TA3 and TC3 are fired. This implied that TA3 and TC3 must be enabled (i.e. Gard4 is satisfied and $(M(PA3).syn \geq 1\ and\ M(PC3) \geq 1)$).

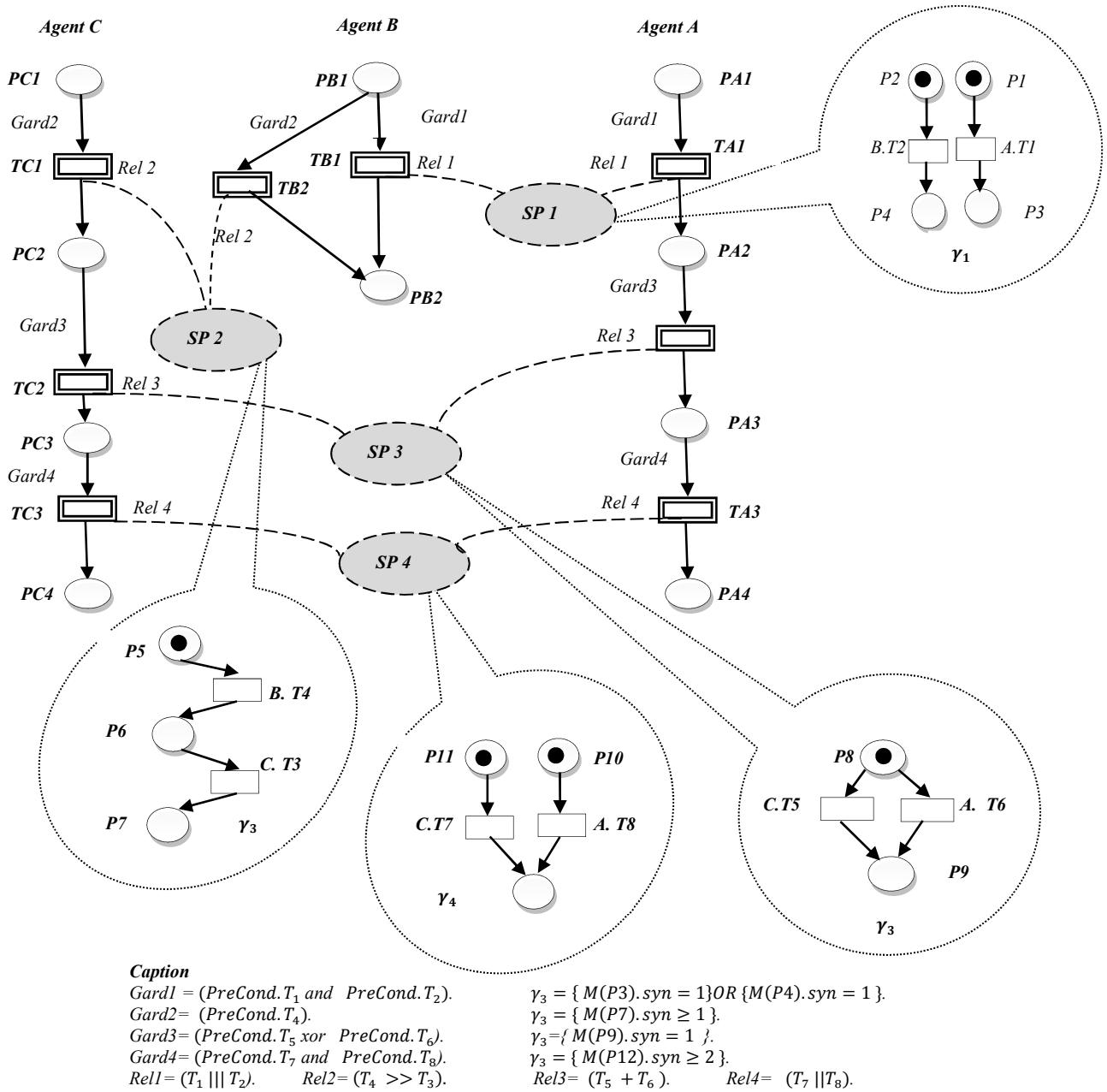
Finally, observe that such a modeling where several nets models each one a special agent, agent synchronization tasks that can run concurrently, is not really handled by standard RPN formalisms. It is probably the reason why in [3], the proposed example dealing with a similar use case, only showed local agent behaviors modeling. In fact, the difficulty was to demonstrate both local and collective behaviors together. (See section 1, paragraph that explains the limitation of the RPN model for specifying synchronization aspects).

B. FIPA Contract Net Interaction Protocol (Concurrency of Interaction)

To enhance the SyPN modeling capabilities when considering dynamic interactions, we now consider the most widely used protocol in MAS area, which is the contract net protocol. In the FIPA standardized version, there are two distinct agent's roles: *initiator* (one agent) and *participants* (numerous instances). For sake of clarity we refer to the Contract-Net's explanation of [21]. A similar case study has been specified by Colored RPN [39]. However, this model allows only the representation of collective behavior whereas SyPN can specify both collective and internal behaviors. In fact SyPN preserves intuition of firing abstract and elementary transitions of RPN, and by the way it allows automatically agents' internal behaviors representation.

"The *initiator* solicits m proposals from other agents by issuing a call for proposals (CFP) act , which specifies the task, as well any conditions the Initiator is placing upon the execution of the task. Agents (*participants*) receiving the call for proposals are viewed as potential contractors and are able to generate n responses. Of these, j are proposals to perform the task, specified as propose acts. The *participant's* proposal includes the preconditions that the *participant* is setting out for the task, which may be the price, time when the task will be done, etc. Alternatively, the $i = n - j$ *participants* may refuse to propose. Once the deadline passes, the *initiator* evaluates the received j proposals and selects agents to perform the task; one, several or no agents may be chosen. The l agents of the selected proposal(s) will be sent an accept-proposal act and the remaining k agents will receive a reject-proposal act . The proposals are binding on the *participant*, so that once the *initiator* accepts the proposal, the *participant* acquires a commitment to perform the task. Once the *participant* has completed the task, it sends a completion message to the *initiator* in the form of an inform-done or a more explanatory version in the form of an inform-result. However, if the *participant* fails to complete the task, a failure message is sent."

We ask the question: how to model such interactions? Let us consider the following scenario: $m = 3, n = j = 3, k = 1$ and $l = 2$. A potential modeling of such scenario is given in Fig. 10. Initially, the *initiator* (I) is ready to send a call for proposal (CFP) to all participants who are ready to interact with it in order to achieve such proposal (here *m* corresponding to Fig 9 is equal to 3): participant 1, participant 2 and participant 3 noted respectively *P, P'* and *P"*. Synchronization conditions corresponding to this remote synchronization are noted *G1, G11P, G11P'* and *G11P"* where *G1* presents task announcement arrival, *G11P, G11P'* and *G11P"* present conditions behind capabilities to achieve the task corresponding to each participant. When *G1, G11P, G11P'* and *G11P"* are fulfilled and $(M(P11).syn \geq 1\ and\ M(PP1).syn \geq 1\ and\ M(P'1).syn \geq 1\ and\ M(P"1).syn \geq 1)$ is satisfied then *T11, TP1, T'1and T"1* are consequently enabled. Concurrent firing of these transitions invokes a rendez-vous; concretized by a synchronization point SP1 (Fig. 11). Once the *participants* receive CFP, they start evaluating the proposal. Based on their capabilities and resources availability, they make decision to perform the proposed task or refuse the request. Note that decisions are made locally (internal behavior, we have made abstraction of such behavior). After making decisions, *P* and *P'* have to prepare their bids that satisfy the criteria specified in the CFP. However, *P"* refuse the request (*i* is equal to 1and *j* is equal to 2; in Fig. 9). Then, *initiator* is waiting for bids or for time out. Synchronization conditions corresponding to second remote synchronization are: *G2= time-expiration (deadline)*, and *G22P, G22P', G22P"* present deliberate-proposal corresponding respectively to *P, P'* and *P"*. when *G22, G22P, G22P'* and *G22P"* are fulfilled and $(M(P12).syn \geq 1 \text{ and } M(PP2).syn \geq 1 \text{ and } M(P'2).syn \geq 1 \text{ and } M(P"2) \geq 1)$ is satisfied then *T12, TP2, T'2* and *T"2* are consequently enabled. The concurrent firing of these transitions invokes a rendez-vous concretized by SP2 (Fig. 11).



Caption

Gard1 = $(PreCond.T_1$ *and* $PreCond.T_2)$.
Gard2= $(PreCond.T_4)$.
Gard3= $(PreCond.T_5$ *xor* $PreCond.T_6)$.
Gard4= $(PreCond.T_7$ *and* $PreCond.T_8)$.
Rel1= $(T_1 ||| T_2)$. *Rel2*= $(T_4 >> T_3)$.

$\gamma_3 = \{ M(P3).syn = 1\} OR \{M(P4).syn = 1 \}$.
$\gamma_3 = \{ M(P7).syn \geq 1 \}$.
$\gamma_3 = \{ M(P9).syn = 1 \}$.
$\gamma_3 = \{ M(P12).syn \geq 2 \}$.
Rel3= $(T_5 + T_6)$. *Rel4*= $(T_7 || T_8)$.

Figure 8. Modeling generic example of multi agent planning by means SyPN.

Then, the *initiator* starts evaluating the received bids and decides to accept the bid corresponding to *P1* and reject the second one (bid of *P'*). *Initiator* have to prepare an accept proposal message to *P1* and a reject proposal message to *P'*. The synchronization point SP3 (Fig. 11) can be invoked when *T13, TP3, T'3* and *T"3* are enabled (i.e. *(G3= G33P = G33P'= true)* and *(M (P13).syn ≥ 1 and M (PP3).syn ≥ 1and M (P'3).syn ≥ 1 and M (P"3).syn ≥ 1)* is fulfilled). *Participant P* has to accomplish the proposed task and prepare the result to be sent to the *initiator*. When (*G4=G44= true)* and *(M (P14).syn ≥ 1 and M (PP4).syn ≥ 1))* is fulfilled then the last rendez-vous for this scenario took place (SP4, Fig. 11). Note that, all send and receive primitives are evaluated during synchronization. For instance, the primitive *SendI1* is evaluated before sending the message as follow: *sender= I, GetReceiver ( ) = {P,P',P"}* and *Msg= CFP* respecting ACL-presentation message. Also, the primitives *RcvP1( ), RcvP'( )* and *RcvP"( )* are evaluated during receiving message. Synchronization points are chosen dynamically by the binding function (variables indicated in the Fig. 9 are evaluated during execution: m=3, j=2, i = k = l =1).

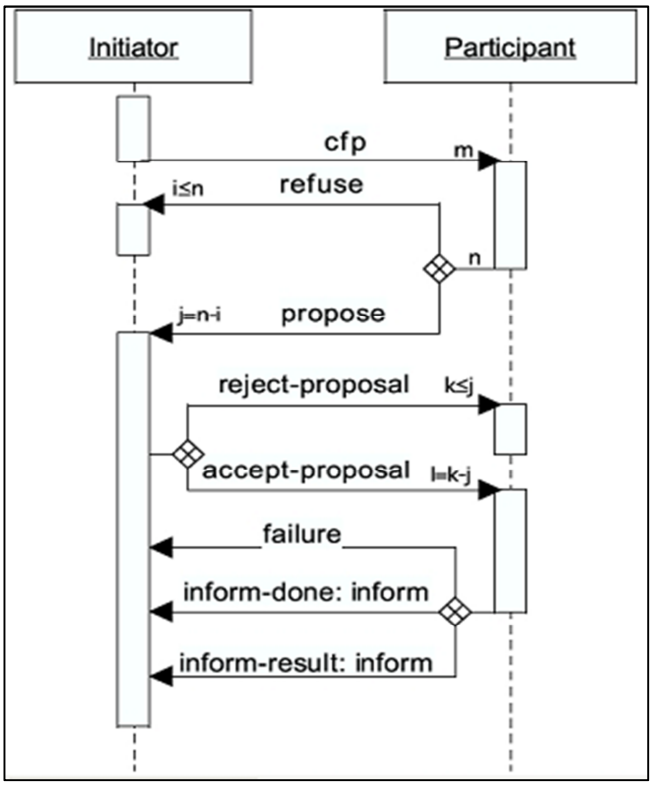


Figure 9. FIPA Contract Net Interaction Protocol.

- SP1 presents sending of CFP by Initiator (*I.T1*) and receiving of this later by *P, P'* and *P"* (*P.T2, P'.T3* and *P".T4*).
- SP2 presents sending of bids, by *P, P'*; sending of reject proposal by *P"* (*P.T5, P'.T6* and *P".T7*), and receiving of bids by *initiator* (*I.T8*).
- SP3 presents sending of response to *P* and *P"* after deliberation (*I.T9*), and receiving of these responses by *P* and *P'* (*P.T10* and *P'.T11*).
- SP4 presents sending of the result after executing the CFP by *P* (*P.T12*) and receiving of this response by the *initiator* (*I.T13*).

Each SPi has an initial marking and a semi linear set of final marking. In our modeling, we have chosen to abstract the internal behavior of agents, to focus on the modeling of concurrent synchronization tasks, however, the modeling can be augmented to deal with internal agent behaviors, without difficulty.

V. Related Work

Several powerful formalisms have been proposed in MAS' specification area. This section reviews some of them:

Z-Specification Language: The Z-Specification Language was developed, initially for formal specification of software systems [46]. It is applied in MAS as well. For instance, d'Inverno et al. [11] have adapted this formalism to construct a formal agent framework. The Z-Specification Language has the following advantages:

- It deals with a clear, precise and unambiguous specification.
- The use of *schemas* and *schemas-inclusion* allows the description of system at different level of abstraction.
- It has a suitable expressiveness that allows a consistent, unified and structured description of a computer system and its associated operations.

As a result of many applications of Z-based framework, it has been argued [11] [43] [51] that Z is inappropriate to model interactions between agents. Also, the use of Z makes MAS reactive aspects difficult to specify and the specifications are not executable.

To address these issues, one can combine two or more formalisms in order to specify easily and naturally the system. For instance, combining Z and CSP, combining Z; object paradigm and State charts [51]. The main criticism of multi formalism is the complexity inherent from this composition. Also, the consistency of such combination is not ensured.

Colored Petri Net: As noted in [16], interactions between agents may be modeled by means of colored Petri net (CPN). Indeed, CPN are suitable for specifying concurrency and the dynamic nature of MAS. Nevertheless this model is very limited when it is necessary to specify the system at different level of abstraction. In other words, CPN does not support abstraction and refinement paradigm, which is a key element of MAS.

Maude: In the same context of specifying interactions between agents, authors in [46] have proposed an approach to formal specification of interactions protocols by means of Maude language.

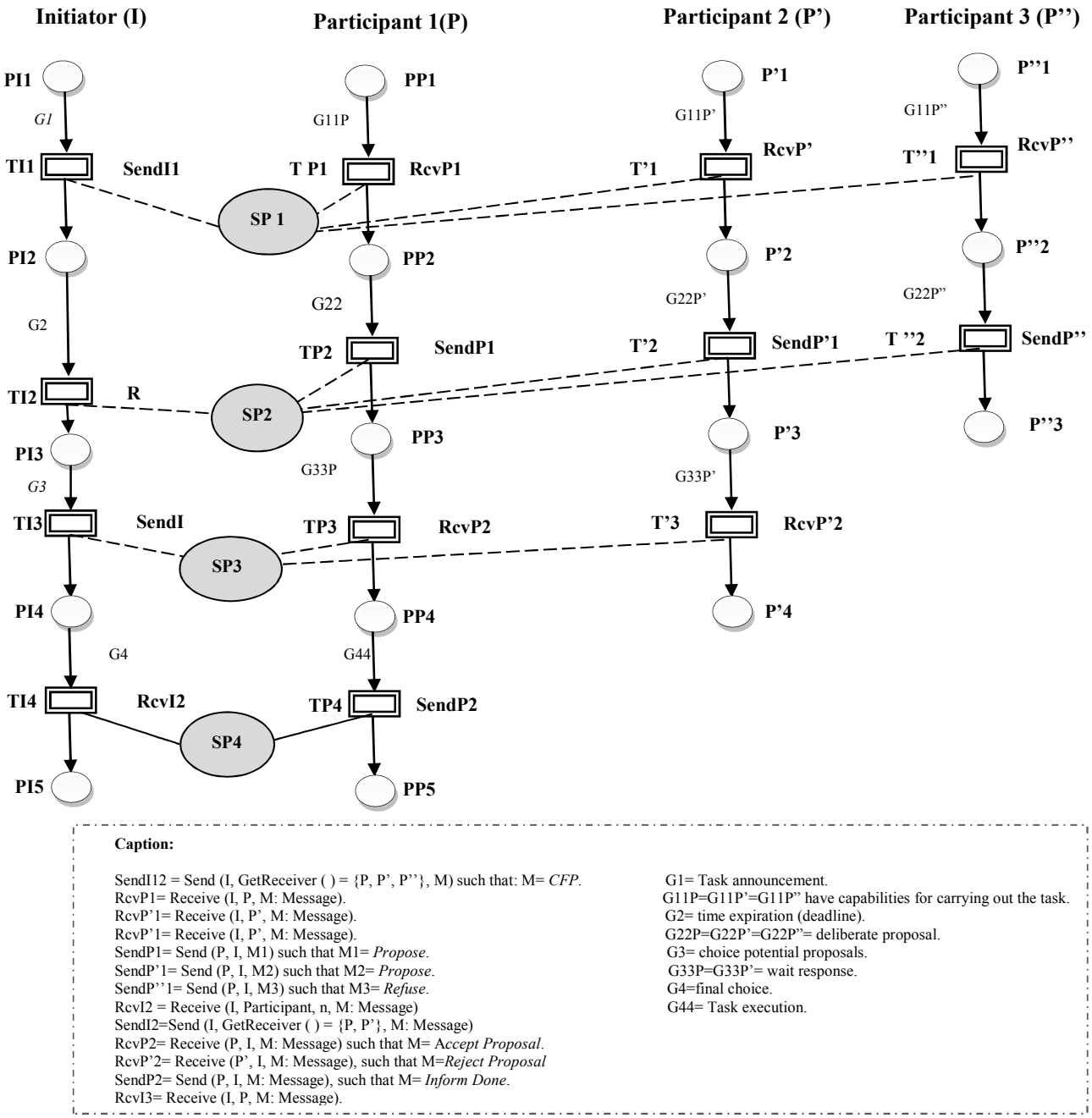


Figure10. Modeling ContractNet Protocol by means of SyPN.

Maude is a specification and programming language, based on rewriting logic. It allows specifying different kinds of concurrent systems. However, Maude cannot deal with asynchronous aspect that represents a fundamental characteristic of MAS.

Logic: Temporal logic has been widely used in MAS area. It is argued that logic provides a precise and unambiguous formal tool to specify and reason about complex systems [40]. However, logic specifications do not provide constructive methodologies for building distributed systems, and so they can be of only limited significance in practice.

These different views have led to divergence of theoretical and practical distributed computing fields [41][42].

As it may be noticed, the above modeling approaches prohibit modeling of at least one of the following characteristics: abstraction and refinement, asynchronous aspects, synchronization between several processes. Anyway SyPN covers all these functionalities. It has the following characteristics:

- **Generality**: SyPN is a general model in the sense that it allows expressing all MAS's

functionalities independently of their applicability or their architecture.

- **Abstraction and refinement**: The concept of abstract transition makes it possible to consider a complex behavior or a refinement task at different levels of abstraction, so one can be free from non-relevant details at a given level of abstraction, hence controlling system's complexity become possible.
- **Synchronization and concurrency**: The concepts of abstract synchronization transition, elementary synchronization transition, synchronization point and synchronization places allows the modeling of synchronization between several nets and thus expressing precisely various mechanisms of rendez-vous (i.e. various interaction's schemas between several agents). The concept of synchronization relation allows one to express clearly and precisely various kinds of synchronization. Concurrency can be easily achieved without extra effort in designing communication and synchronization mechanism because SyPN expresses them implicitly.
- **Preservation of agent's properties**: The token typing preserves agents' proprieties such as autonomy, awareness (i.e. local token) and sociability (i.e. synchronization token).
- **Dynamicity**: The dynamic nature of MAS can be easily reflected by SyPN's features such as non-determinism, abstraction, etc.
- Creation and destruction of agents can be easily modeled by a transition.

Although, the above arguments show the interests of the SyPN model for specifying MAS, a formal comparison against existing models remain to be done.

VI. CONCLUSION

In this paper, we presented a recursive Petri net based formal specification model covering various functionalities of MAS. We enriched RPN by several features enabling among others modeling both internal and collective behaviors of MAS. Firstly, we presented different behaviors of MAS. Secondly we defined syntax and formal semantic of the model; either we showed intuition behind all novel concepts. Thirdly, we clarified the effectiveness and expressiveness of SyPN through two examples. The former one showed SyPN's efficiency for rigorous specification of multi agent planning. It confirmed the significance of interaction operators. The second one gave suitability of SyPN to remote interactions modeling. Consequently, we saw how SyPN can specify both local synchronization and remote interaction. Fourthly, we discussed roughly several existing formalisms in MAS specification area and explained their inadequacy to rigorously specifying main features of MAS. Moreover, we provided numerous gained advantages for SyPN.

This work may be continued in several ways. Firstly, we investigate this model for proposing a design methodology for MAS based on SyPN specification model. Also, the automatic generation of the reachability graph allows using existing formal checking tools. Other alternatives are: investigating mobility feature of SyPN, through a concrete example and defining a formal semantic of true concurrency [53] for SyPN.

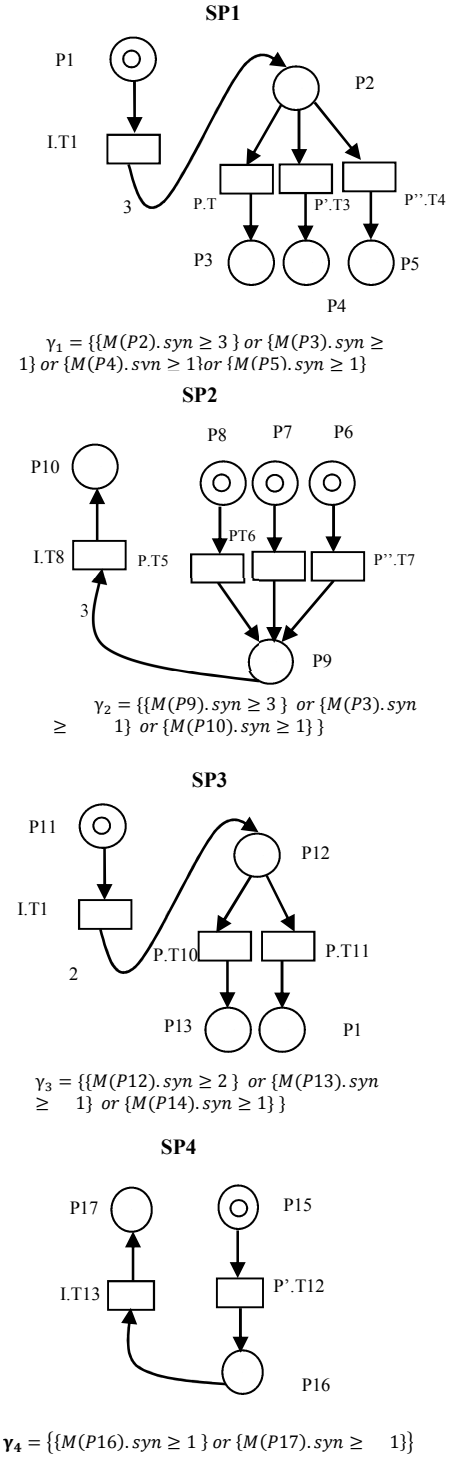


Figure 11. Synchronization Points associated to Fig.10.

ACKNOWLEDGMENT

The authors wish to acknowledge gratefully the valuable suggestion and beneficial comments provided by the anonymous reviewers.

REFERENCES

[1] F. Brazier, B. Dunin Keplicz, N. Jenning and J. Treur. “Desire: Modeling Multi-Agent Systems in a Compositional Formal Framework”. International Journal of Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems. Pp.67-94. 1997.

[2] M. Bettaz , M Maouche , K. Barkaoui. “Formal Specification of Communication Protocols with Object-Based ECATNets”. 22nd EUROMICRO Conference, pp.492 - 499, January 1996.

[3] S. Boussetta. "Stratégies d’exécution de Plans d’un Système Multiagent". PhD thesis, Université de Paris 09, Paris, FRANCE, 1998.

[4] S. Boussetta, A. El Fallah Seghrouchni , S. Haddad, P. Moraitis and M. Taghelit. “Coordination d'agents Rationnels par Planification Distribuée". In RIA (Revue d'Intelligence Artificielle). Edition Hermès. 1998.

[5] G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez-Sanz, J. Pavon, P. Kerney, J. Stark, and P. Massonnet. “Agent Oriented Analysis using Message/Uml”. Second International Workshop, AOSE 2001, Montreal, Canada, May 29, 2001. Revised Papers and Invited Contributions. LNCS vol. 2222 (May 2001). pp. 119-135. Springer-Verlag. 2002.

[6] J. R. Celaya, A. A. Desrochers and R. J. Graves. “Modeling and Analysis of Multi-agent Systems using Petri Nets”. Journal of Computers, VOL. 4, NO. 10, OCTOBER 2009.

[7] P. R. Cohen and H. J. Levesque. “Intention is Choice with Commitment”. Artificial Intelligence, vol. 42.pp. 213-261. 1990.

[8] D. Dahmani, J-M. Ilié and M. Boukala. “Reachability Analysis for Recursive Petri Nets with Shared Places". In the International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC) Paris, France, June 2009.

[9] S. A. DeLoach, M.F. Wood and C. H. Sparkman. “Multi-agent Systems Engineering”. International Journal of Software Engineering and Knowledge Engineering. Vol. 11. Issue 3. pp. 231-258. 2001.

[10] Y. Demazeau. “From Interactions to Collective Behaviour in Agent-based Systems”. In Proceedings of the First European Conference on Cognitive Science, Saint Malo, France. Pp.117-132. 1995.

[11] M. d’Inverno, M. Fisher, A. Lomuscio, M. Luck, M. Rijke, M. Ryan and M. Wooldridge. “Formalisms for Multi-Agent Systems”. Lecture Notes in the First UK Workshop on Foundations of Multi-Agent Systems (Fo-MAS’96).http://citeseer.nj.nec.com.

[12] J. Dix,E. H. Durfee, and C. Witteveen. “Planning in Multiagent Systems”. Dagstuhl Seminar Proceedings 08461. 2009. http://drops.dagstuhl.de/opus/volltexte/2009/1874.

[13] A. El Fallah Seghrouchni “Rational Agent Cooperation through Concurrent Plan Coordination”. In the proceedings of DAIMAS'96 (First Iberoamerican Workshop on DAI and MAS). Xalapa, Mexico, 1996.

[14] A. El Fallah Seghrouchni and S. Haddad. “A Coordination Algorithm for Multi-Agent Planning”. In MAAMAW'96, LNAI 1038, pp. 86-99. Springer, 1996.

[15] A. El Fallah Seghrouchni and S. Haddad. “A Recursive Model for Distributed Planning”. In M. Tokoro (Ed.), Proceeding Second International Conference on Multi-Agent Systems.ICMAS'96, AAAI Press (1996). pp. 307-314.1996.

[16] A. El Fallah Seghrouchni, S. Haddad and H. Mazouzi: “Protocol Engineering for Multi-Agents Interaction”. In Multi-Agent System Engineering, 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW '99, LNAI 1647, pp.89-101. Springer, 1999.

[17] J. Ferber, and O. Gutknecht. “A Meta-Model For the Analysis and Design of Organizations in Multi-Agent Systems” In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS’98), IEEE Computer Society Press, pp. 128–135. 1998.

[18] T. Finin, R. Fritzson, D. McKay and R. McEntire. “KQML as an Agent Communication Language”. CIKM '94 Proceedings of the third international conference on Information and knowledge management, pp.456 – 463. 1994..

[19] M. Fisher. “A Survey of Concurrent Metatem The Language And Its Applications, Temporal Logic”. In Proceedings of the First International Conference, LNCS, vol.82.pp 480-505., Springer Verlag 1994.

[20] Foundation for Intelligent Physical Agents. http://www.fipa.org/

[21] Foundation for intelligent physical agents: FIPA Contract Net Interaction Protocol. Standard. 2002.

[22] N. Glaser. “Contribution to Knowledge Modeling in A Multi-Agent Framework (The Comomas Approach)”, Ph.D thesis. University of Henri Poincare, Nancy I. 1996.

[23] B. J.Grosz and S. Kraus. “Collaborative Plans for Complex Group Actions”. Artificial Intelligence, vol. 86. pp. 269-357. 1996.

[24] S. Haddad and D. Poitrenaud. “Checking Linear Temporal Formulas on Sequential Recursive Petri Nets". In TIME'01, pp. 198-205. IEEE Computer Society Press, 2001.

[25] S. Haddad and D. Poitrenaud “Modeling and Analyzing Systems with Recursive Petri Nets”. In WODES'00, pp. 449-458. Kluwer Academic Publishers. 2000.

[26] S. Haddad and D. Poitrenaud “Recursive Petri Nets, an Expressive Model For Discrete Event Systems”. Acta Informatica 44(7-8), pages 463-508. 2007.

[27] S. Haddad and D. Poitrenaud. “Theoretical Aspects of Recursive Petri Nets” In ICATPN'99, LNCS, vol. 1639, pp. 228-247. Springer. 1999.

[28] Z .Haibin and Z. Meng Chu. “Role-based Multi-Agent Systems”. Personalized Information Retrieval and Access: Concepts, Methods and Practices. Chapter 12.pp. 254-285. 2008.

[29] Hang-Jiang Gao , Zheng Qin, Lei Lu, Li-Ping Shao and Xing-Chen Heng. “Formal Specification and Proof of Multi-Agent Applications Using Event B”. Information Technology Journal. Vol. 6 issue 8, Pp. 1181-1189. 2007.

[30] Z. Hou, Z. Yu, W. Zheng and X. Zuo. "Research on Distributed Intrusion Detection System Based on Mobile Agent" Journal of Computers, Vol. 7, No. 8, pp. 1919-1926. August 2012. “doi:10.4304/jcp.7.8.1919-1926".

[31] L. Jemni Ben Ayed. Ayed and F. Siala. “Event-B based Verification of Interaction Properties. In Multi-Agent Systems”. Journal of Software, Vol. 4, NO. 4, 2009.
[32] L. Jemni Ben Ayed, and F. Siala, “Specification and Verification of Multi-Agent Systems Interaction Protocols using a Combination of AUML and Event B” XVth International Workshop on the Design, Verification and Specification of Interactive Systems DSV-IS 2008, LNCS 5136, Kingston, Ontario, Canada, pp. 102-107. 2008.
[33] N. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra and M. Wooldridge. “Automated Negotiation: Prospects, Methods and Challenges”. In International Journal of Group Decision and Negotiation. 10(2), pages 199-215, 2001.N.
[34] Jennings and M. Wooldridge. “Applications of Intelligent Agents”. In Agent Technology: Foundations, Applications, and Markets. Springer, pp. 3-28.1998.
[35] N. R. Jennings and M. Wooldridge. “Software Agents". IEE Review Vol. 42, issue 1, pp. 17-21. January 1996.
[36] S. J. Juneidi, G. A. Vouros. “Agent Role Locking (ARL): Theory for Multi Agent System with E-Learning Case Study”. IADIS AC 2005: pp.442-450.2005.
[37] F. Kabanza, "Synchronizing Multi Agent Plans using Temporal Logic Specifications”. V. Lesser (Ed.), Proc. First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA (1995), pp. 217–224. 1995.
[38] D. Kinny, M. Ljungberg, A. S. Rao, E. A. Sonenberg, G. Tidhar and E. Werner. “Planned Team Activity”. Proceedings of the fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World. (MAAMAW'92). 1992.
[39] Y. Kissoum, Z. Sahnoun: “A Recursive Colored Petri Nets Semantics for AUML as base of Test Case Generation”. The 6th ACS/IEEE International Conference on Computer Systems and Applications, Doha, Qatar. Pp.785-792. 2008.
[40] J. Lind “Iterative Software Engineering for Multi-Agent Systems: The Massive Method” Springer 2001.
[41] A. Lomuscio, M. Sergot "Deontic Interpreted Systems”. Studia Logica. Vol. 75, issue 1. Pp. 63-92. 2003.
[42] A. Lomuscio and M. Sergot. “On Multi-agent Systems Specification via Deontic Logic”. Proceedings of ATAL01, Agent Theories Languages, and Architectures. Seattle, August. Springer Verlag Lecture Notes in AI vol. 2333 (Intelligent Agents VIII).2001.
[43] M. Luck, N. Griffiths and M. d’Inverno. “From Agent Theory to Agent Construction: A Case Study”. In Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, J. P. Muller, M. J. Wooldridge and N. R. Jennings (eds.), Lecture Notes in Artificial Intelligence, 1193, 49–63, Springer-Verlag, 1997.
[44] V. Marik, J. Müller and M. Pechoucek. “Multi-agent Systems And Applications II”, LNAI 2691, pp. 394-403.Springer-Verlag 2003.
[45] H. Mazouzi, A. El Fallah Seghrouchni and S. Haddad. "Open Protocol Design for Complex Interaction in Multi-Agent Systems". In AAMAS'02, pages 15-19. ACM Press, 2002.
[46] F. Mokhati, N. Boudiaf, M. Badri and L. Badri “Translating AUML Diagrams into Maude Specifications: A Formal Verification of Agents Interaction Protocols”. Appear in Journal of Object Technology, Vol. 6, No 4. pp. 77- 102. 2007.
[47] P. Singh Munindar , S. Rao Anand, and P. Michael Georgeff. “Formal Methods in DAI: Logic-Based Representation and Reasoning”. Gerhard Weiss (ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, chapter 8, pp. 331–376. 1999.
[48] J. Pavón , J.J. Gómez-Sanz and R. Fuentes. “Model Driven Development of Multi-Agent Systems”. In European Conference on Model Driven Architecture. LNCS. Vol. 4066, Pp. 284-298. Springer, Heidelberg 2006.
[49] I. Pereverzeva, E. Troubitsyna and L. Laibinis. “Formal Goal-Oriented Development of Resilient MAS in Event-B”. TUCS Technical Report. No 1033, January 2012.
[50] A. Regayeg, A.H. Kacem, and M. Jmaiel, “Specification and Verification of Multi Agent Applications Using Temporal Z” In Intelligent Agent Technology Conference (IAT.04), IEEE Computer Society, pp.260-266. 2004.
[51] S. Rodriguez, V. Hilaire and A. Koukam. “Formal Specification of Holonic Multi-Agent Systems Framework”. In Vaidy S. Sunderam, Geert Dick van Albada, Peter M. A. Sllot, and Jack J. Dongarra, editors, Computational Science - ICCS 2005, number 3516 in Lecture Notes in Computer Science, pages 719 – 726. Springer, 2005.
[52] Y. Al. Saawy, A. Al-Ajlan; K. Aldrawiesh and A. Bajahzer, A. “The Development of Multi-agent System Using Finite State Machine”. Appear in New Trends in Information and Service Science. NISS '09. Pp. 203 – 206. 2009.
[53] D.E. Saïdouni, N. Belala, and M. Bouneb. “Aggregation of Transitions in Marking Graph Generation Based on Maximality Semantics for Petri Nets” In Proceedings of the Second International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS’2008), University of Leeds, UK. eWiC Series, The British Computer Society (BCS),July, 2-3rd 2008. ISSN: 1477-9358.
[54] Supriya D'Souza, Abhishek Rao, Amit Sharma and Sanjay Singh. “Modeling and Verification of a Multi-Agent Argumentation System using NuSMV”. arXiv:1209.4330,v1 [cs.AI] 2012.
[55] M. Wang, Z. Shi and W. Jiao. “Dynamic Interaction Protocol Load in Multi-agent System Collaboration”. Multi-Agent Systems for Society. Lecture Notes in Computer Science. Vol. 4078, pp. 103-113. 2009.
[56] M. Wooldridge and N. R. Jennings. “Intelligent Agents: Theory and Practice”. In Knowledge Engineering Review Vol.10, issue 2, pp. 115-152.1995.
[57] Y. Xu and X. Xie. “Modeling and Analysis of Security Protocols Using Colored Petri Nets” Journal of Computers. Vol. 6, No. 1, pp 19-27. JANUARY 2011. “Doi:10.4304/Jcp.6.1.19-27”.