Theoretical Aspects of Recursive Petri Nets

Serge Haddad[1] and Denis Poitrenaud[2]

[1] LAMSADE - UPRESA 7024, Université Paris IX, Dauphine
Place du Maréchal Delattre de Tassigny, 75775 Paris cedex 16
[2] LIP6 - UMR 7606, Université Paris VI, Jussieu
4, Place Jussieu, 75252 Paris cedex 05

Abstract. The model of recursive Petri nets (RPNs) has been introduced in the field of multi-agent systems in order to model flexible plans for agents. In this paper we focus on some theoretical aspects of RPNs. More precisely, we show that this model is a strict extension of the model of Petri nets in the following sense : the family of languages of RPNs strictly includes the union of Petri net and Context Free languages. Then we prove the main result of this work, the decidability of the reachability problem for RPNs.

1 Introduction

Since the introduction of Petri nets, even before the decidability of the reachability problem has been solved, theoretical works have been developed in order to study the impact of extensions of Petri nets on this problem. For instance, the reachability problem is undecidable for Petri nets with two inhibitor arcs while it becomes decidable with one inhibitor arc or a nested structure of inhibitor arcs [Rei95]. The self-modifying nets introduced by R. Valk have (like Petri nets with inhibitor arcs) the power of Turing machine and thus many properties including reachability are undecidable [Val78a,Val78b]. Introducing restrictions on self-modifying nets enables to decide some properties [DFS98] (boundedness, coverability, termination,...) but the reachability remains undecidable.

Recently Recursive Petri nets (RPNs) have been proposed for modeling plans of agents in a multi-agent system [SH95,SH96]. A RPN has the same structure as an ordinary one except that the transitions are partitioned into three categories : elementary transitions, abstract transitions and final transitions. Moreover a starting marking is associated to each abstract transition. The semantics of such a net may be informally explained as follows. In an ordinary net, a thread plays the token game by firing a transition and updating the current marking (its internal state). In a RPN there is a dynamical tree of threads (denoting the fatherhood relation) where each thread plays its own token game. The step of a RPN is thus a step of one of its threads. If the thread fires an elementary transition, then it updates its current marking using the ordinary firing rule. If the thread fires an abstract transition, it consumes the input tokens of the transition and generates a new child which begins its token game with the starting marking of the transition. If the thread fires a final transition, it aborts its whole descent of threads, produces (in the token game of its father) the output tokens of the abstract transition which gave birth to it and dies. In case of the root thread, one obtains an empty tree.

The modeling capabilities of RPNs can be illustrated in different ways. In a subsequent section, we present the modeling of faults within a system whereas an equivalent modeling by an ordinary Petri net is not known to be possible. RPNs enable also to easily model multi-level executions (e.g. interruptions). In case of planning of agents, abstract transitions model differed plannings of complex actions.

As soon as a new model is proposed, a significant question (at least if a semantics of the firing sequence is possible) is to know whether the model is really an extension or simply an abbreviation. For instance the model of colored Petri nets with finite color domains is an abbreviation whereas allowing infinite domains extends the ordinary Petri nets. In case of RPNs we show that this model is a strict extension of ordinary Petri nets. Our proof is based on a result in [Jan79] which establishes that the palindrome language is not a Petri net language (even in the largest definitions). Thus we exhibit a RPN which recognizes this language. Moreover, we prove that RPN languages strictly include the union of Context Free and Petri net languages.

Then we tackle with the main question about strict extensions of Petri nets: does the reachability problem remain decidable ? This question is theoretically important as it seems a limit result. Recently in [Rei95], it has been stated that reachability is decidable for Petri nets with one inhibitor arc (or more generally with a nested structure of inhibitor arcs). We prove that the reachability problem is also decidable for RPNs. Our proof is divided into three steps. First we prove that one can decide whether the thread initiated by an abstract transition can die by some sequence in which case we call such a transition a "closable transition". Then we study the structure of a hypothetical sequence for the reachability and show that its existence is equivalent to the existence of sequences of reachability for initial and final states of less complexity (in terms of the size of their tree of threads). Finally we show that when the initial and final states are associated to one (or zero) thread, then the problem is equivalent to the ordinary reachability where the non closable abstract transitions are deleted and each closable one is replaced by an equivalent elementary one.

The balance of the paper is the following. In Sect. 2, we define the RPNs, we give some examples in order to understand their behavior and their modeling capability. Then, we show that the model is a strict extension of Petri nets and that their languages strictly include the union of Petri net and Context Free languages. In Sect. 3, we prove the decidability of the reachability problem. In the last section, we conclude giving some perspectives to this work.

2 Recursive Petri Nets

2.1 Structure

A *recursive Petri net* is defined by a tuple $N = \langle P, T, W^-, W^+, \Omega \rangle$ where

- P is a finite set of places.
- T is a finite set of transitions.
- A transition of T can be either elementary, abstract or final. The sets of elementary, abstract and final transitions are respectively denoted by T_{el}, T_{ab} and T_{fi} (with $T = T_{el} \uplus T_{ab} \uplus T_{fi}$ where $\uplus$ denotes the disjoint union).
- W^- and W^+ are the pre and post flow functions defined from $P \times T$ to $\mathbb{N}$.
- Ω is a labeling function which associates to each abstract transition an ordinary marking (i.e. an element of $\mathbb{N}^P$).

An *extended marking* tr of a recursive Petri net $N = \langle P, T, W^-, W^+, \Omega \rangle$ is a labeled tree $tr = \langle V, M, E, A \rangle$ where V is the set of vertices, M is a mapping $V \to \mathbb{N}^P$, $E \subseteq V \times V$ is the set of edges and A is a mapping $E \to T_{ab}$. We denote by $v_0(tr)$ the root node of the extended marking tr. The edges E build a tree i.e. for each v different from $v_0(tr)$ there is one and only one $(v', v) \in E$ and there is no $(v, v_0(tr)) \in E$. Any ordinary marking can be seen as an extended marking composed by a unique node. The empty tree is denoted by $\perp$.

Remark: An extended marking does not depend on V the set of vertices. Given two extended markings, if there is a one-to-one mapping between the two sets of markings which preserves the set of edges, the labeling of vertices and of the edges then the two markings are equal.

A *marked recursive Petri net* (N, tr_0) is a recursive net N associated to an initial extended marking tr_0. This initial extended marking is usually a tree reduced to a unique vertex.

For a vertex v of an extended marking, we denote by $pred(v)$ its (unique) predecessor in the tree (defined only if v is different from the root) and by $Succ(v)$ the set of its direct and indirect successors including v ($\forall v \in V, Succ(v) = \{v' \in V \mid (v, v') \in E^*\}$ where E^* denotes the reflexive and transitive closure of E).

A *branch* br of an extended marking tr is one of the subtrees rooted at a son of $v_0(tr)$. One can associate to a branch a couple (t, tr) where t is the abstract transition which labels the edge leading to the subtree and tr the subtree taken in isolation. Let us note that the couple (t, tr) characterizes a branch.

In other words, given an extended marking tr, a branch br with its couple (t, tr') fulfills : tr' is a sub-tree of tr verifying $(v_0(tr), v_0(tr')) \in E$ (i.e. in tr, the root of tr' is a direct successor of the root of tr) and $A(v_0(tr), v_0(tr')) = t$ (i.e. in tr, the arc between the root of tr and tr' is labeled by t).

We denote by $Branch(tr)$ the set of branches of an extended marking tr and by $branch(tr, t)$ the subset of Branch(tr) where the edge leading to the subtree is labeled by t.

We denote by (m, Br), where m is an ordinary marking and Br a set of branches, the extended marking tr verifying $M(v_0(tr)) = m \wedge Branch(tr) = Br$.

The depth of an extended marking is recursively defined as follows : let m be an ordinary marking and tr an extended marking.

- $depth(\perp) = 0$
- $depth(m) = 1$
- $depth(tr) = max(\{depth(tr') \mid \exists (t, tr') \in Branch(tr)\}) + 1$

2.2 Semantics

A transition t is enabled in a vertex v of an extended marking tr iff $\forall p \in P, M(v)(p) \geq W^-(p,t)$. In other words, the thread associated to each node uses the same rule for the enabling of a transition as for ordinary Petri nets.

We denote by $tr\xrightarrow{t}$ that there exists a node v of tr in which the transition t is firable.

The firing of an enabled transition t from a vertex v of an extended marking $tr = \langle V, M, E, A\rangle$ leads to the extended marking $tr' = \langle V', M', E', A'\rangle$ depending on the type of t:

***t* is an *elementary transition* ($t \in T_{el}$).** The thread associated to v fires such a transition as for ordinary Petri nets. The structure of the tree is unchanged. Only the current marking of v is updated.

- $V' = V$
- $\forall v' \in V \setminus \{v\}, M'(v') = M(v')$
  $\forall p \in P, M'(v)(p) = M(v)(p) - W^-(p,t) + W^+(p,t)$
- $E' = E$
- $\forall e \in E, A'(e) = A(e)$

***t* is an *abstract transition* ($t \in T_{ab}$).** The thread associated to v consumes the input tokens of t. It generates a new thread v' with initial marking the starting marking of t. Let us note that the identifier v' is a fresh identifier absent in V.

- $V' = V \cup \{v'\}$
- $\forall v'' \in V \setminus \{v\}, M'(v'') = M(v'')$
  $\forall p \in P, M'(v)(p) = M(v)(p) - W^-(p,t)$
  $M'(v') = \Omega(t)$
- $E' = E \cup \{(v, v')\}$
- $\forall e \in E, A'(e) = A(e)$
  $A'((v, v')) = t$

***t* is a *final transition* ($t \in T_{fi}$).** If the thread is associated to the root of the tree, the firing leads to the empty tree. In the other case, the thread associated to v produces the output tokens of the abstract transition which gave birth to it, in the marking of its father Then it (and its whole descent) dies.

- $V' = V \setminus Succ(v)$
- $\forall v' \in V' \setminus \{pred(v)\}, M'(v') = M(v')$
  $\forall p \in P, M'(pred(v))(p) = M(pred(v))(p) + W^+(p, A(pred(v), v))$
- $E' = E \cap (V' \times V')$
- $\forall e \in E', A'(e) = A(e)$

We denote by $tr \xrightarrow{t} tr'$ that there exists a node v of tr such that the firing of t in v leads the net to the extended marking tr'.

A firing sequence is usually defined : a transition sequence $\sigma = t_0 t_1 t_2 \ldots t_n$ is enabled from an extended marking tr_0 (denoted by $tr_0 \xrightarrow{\sigma}$) iff there exists tr_1, tr_2, …, tr_n such that $tr_{i-1} \xrightarrow{t_i} tr_i$ for $i \in [1, n]$.

Important remark: In a firing sequence, we impose (w.l.o.g.) that any fresh identifier is new not only in the current tree but also in all the previous trees of the sequence. Such a restriction ensures that if the roots of two branches of two trees of the sequence are associated to the same identifier then they denote the same branch - with possible change of structure and marking.

We denote by $\mathcal{L}(N, tr_0, Tr_f)$ (where Tr_f is a finite marking set) the set of firing sequences of N from tr_0 to a marking of Tr_f. This set is called the language of N.

Let σ be a firing sequence and tr_1, tr_2, …, tr_n the extended markings visited by σ, the depth of σ (denoted by $Depth(\sigma)$) is the maximal depth of tr_1, tr_2, …, tr_n.

2.3 Recursive Petri Nets Versus Petri Nets

In this section, we show that the recursive Petri net model is a strict extension of Petri nets.

For this demonstration, we consider *labeled recursive Petri nets.* Such a net associates to a recursive net, a labeling function h defined from the transition set T to an alphabet Σ plus λ (the empty word). h is extended to sequences and also to languages. The language of a labeled recursive Petri net is defined by $h(\mathcal{L}(N, tr_0, Tr_f))$.

The Fig. 1 gives a marked labeled recursive Petri net. The marking associated to both abstract transitions is P.

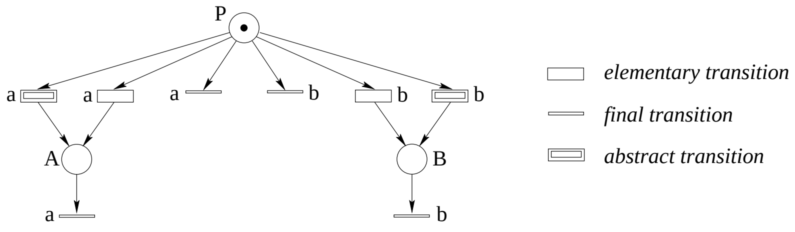

Fig. 1. recursive Petri net which generates palindromes on $\{a, b\}$

Figure 2 presents a prefix of its reachability graph. Notice that the complete graph is infinite. From the initial marking, all the transitions adjacent to the place P are enabled. The firing of the abstract transition labeled a consumes the token of the place P and leads to the construction of a successor node associated to the marking P. The arc between the two nodes is labeled by an a. The firing of the final transition labeled a from the initial marking leads to the empty tree.

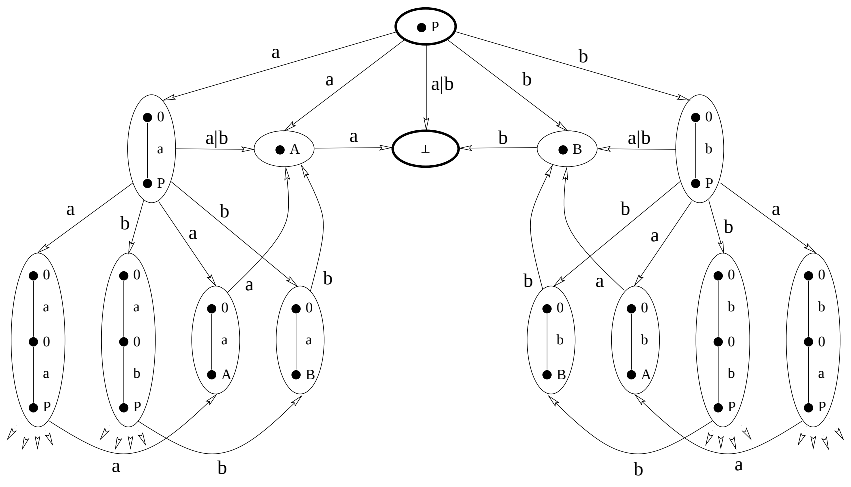

Fig. 2. a prefix of its reachability graph

M. Jantzen has shown in [Jan79] (see also [Lam88]) that $PAL(\Sigma)$ (for $|\Sigma| \geq 2$) is not a Petri net language (allowing λ-transitions).

We consider the language of the recursive net of the Fig. 1 defined by the set of ending markings $\{\perp, P\}$ (the two corresponding markings are represented in bold in the Fig. 2). This language is exactly $PAL(\{a, b\})$ which demonstrates that recursive Petri net model is a strict extension of the Petri net one.

2.4 Expressive Power of Recursive Petri Nets

Indeed, we show that recursive Petri net languages strictly include the union of Context Free and Petri net languages.

Proposition 1 (Strict inclusion). *Recursive Petri net languages strictly include the union of Context Free and Petri net languages.*

Proof (Sketch). Like the complete proof is not particularly difficult, we only give a sketch of this proof based on simple example.

It is clear that recursive Petri net languages include Petri net language. Then, we begin by the inclusion of Context Free languages. We consider a Context Free language defined by a set of symbols partionned in terminal symbols T and non-terminal symbols N. To each non-terminal symbol $s \in N$ is associated a non-empty set $\{s_1, s_2, \ldots, s_n\}$ of words on $T \cup N$. A particular non-terminal symbol s_0 is designated as the initial one. A word of the language is obtained by choosing word in the set associated to s_0 and then by iteratively substituting in it a non-terminal symbol by an element of the associated set until the word does not contain any non-terminal symbol.

For a given Context Free language, we can construct a RPN having exactly the same language.

For any set associated to a non-terminal symbol s, we build a particular subnet of the RPN. First, we add a place P_s. Each element of the set associated to s leads to the production of a sequence of transitions. Let s_i be such an element. For each non-terminal symbol r composing s_i, we add an abstract transition labeled by λ and associated to the ordinary marking P_r. For each terminal symbol a, we add an ordinary transition labeled by a. Moreover, to each of these transitions we add an output place and give as input place the output place of the transition associated to the symbol preceding the considered one in the word s_i. Finally, the input place of the first transition is designated to be P_s and a final transition labeled λ is constructed at the output of the last place in the sequence.

The initial extended marking is composed by only one node for which the ordinary marking is P_{s_0} and the language that we consider is $\mathcal{L}(N, tr_0, \{\perp\})$. It can be shown that this language is exactly the considered Context Free language.

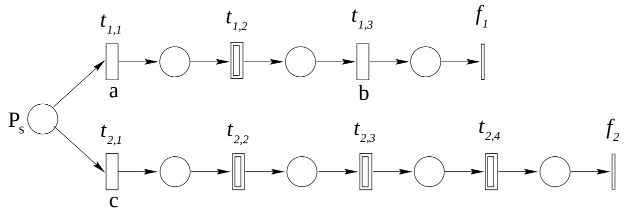

Fig. 3. a RPN modeling a Context Free language

The Fig. 3 illustrates this construction. The part of the RPN corresponding to a non-terminal symbol S associated to the set $\{aSb, cDEF\}$ is given. Symbols in uppercase are considered as non-terminal and the others as terminal. A transition associated to the j^{th} symbol of the i^{th} element of the set is designated by $t_{i,j}$ and the final transition is denoted f_i. As an example, the abstract transition associated to E is denoted $t_{2,3}$. The label λ of the abstract and final transitions has been omitted.

To demonstrate that this inclusion is strict, we present in Fig. 3 (added to the net of the Fig. 1) a RPN for which its language is neither a PN one nor a Context Free one. The ordinary marking associated to the abstract transition t'_0 is the place P of the RPN in the Fig. 1. Moreover, its initial extended marking is composed by a unique node associated to the ordinary marking P_{init}. We consider the language $\mathcal{L}(N, tr_0, \{tr_f\})$ where tr_f is the extended marking composed by only one node associated to the empty ordinary marking. This language is exactly $\{w_1, w_2\}$ where $w_1 \in Pal(\{a, b\})$ and $w_2 \in \{c^n.d^n.e^n\}$ with $n \geq 0$. This

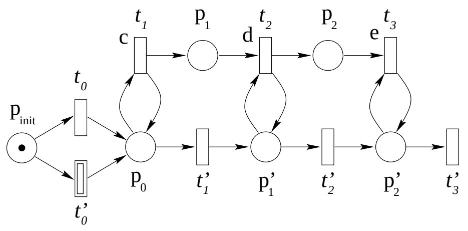

Fig. 4. a particular RPN

language is not a Petri net one (its projection on $\{a, b\}$ gives $Pal(\{a, b\})$) and is not a Context Free one (its projection on $\{c, d, e\}$ gives $\{c^n.d^n.e^n\}$ with $n \geq 0$) which concludes the proof. $\square$

2.5 Recursive Petri Nets and the Modeling of Faults

In order to give insight to the improvements brought by RPNs, we present in the Fig. 5 an easy way to add faulty behaviors to a Petri net model. The required faulty behavior cleans up the current marking and restarts with the initial marking. To the best of our knowledge, there is no general way to add such a feature to a Petri net. Only ad-hoc mechanisms are proposed for particular cases.

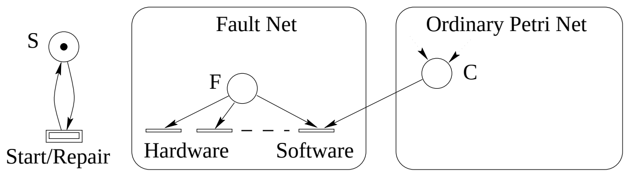

Fig. 5. recursive Petri net modeling faults

Let us look at our figure. The net is divided into three parts. On the right there is the Petri net of the correct behavior (in our formalism, all transitions are elementary). On the center there is the faulty behavior with some spontaneous faults (transition `Hardware`) and some conditioned faults (transition `Software`). For sakeness of the modeling, a control place `F` is added ; all the transitions of this part are final. On the left there is a simple initially marked loop with an abstract transition. The starting marking of this transition is the marking of the Petri net model of the correct behavior plus one token in the control place `F`.

The behavior of the net may be described as follows. Initially and in all the crash states, the extended marking is reduced to one node with the loop place `S` marked. When the abstract transition is fired the correct behavior is "played" by the new thread. If this thread dies by the firing of a faulty transition, we come back to the initial state and so on.

3 Decidability of the Reachability Problem

The reachability problem consists to determine if a given state is reachable from another one.

This problem has been demonstrated to be decidable for ordinary Petri nets (see [May81,Kos82,Lam88]). The main result of this paper is that the reachability problem can be also decided for recursive Petri nets.

3.1 Branches Structure of a Firing Sequence

First, we characterize the different kinds of behaviors of branches inside a sequence.

Definition 2 (Permanent branch). Let tr, tr' be two extended markings and σ be a firing sequence from tr to tr'.
A couple of branches $((t_a, tr_a), (t_b, tr_b)) \in Branch(tr) \times Branch(tr')$ denotes a *permanent branch* in σ if $v_0(tr_a) = v_0(tr_b)$.

The previous definition expresses that the node $v_0(tr_a)$ is never removed by the firing of a final transition in $v_0(tr_a)$. Remark that in this case, we have necessary $t_a = t_b$.

If a branch is not permanent and occurs in the final marking then it has been "opened" by an abstract transition.

Definition 3 (Opened branch). Let tr, tr' be two extended markings and σ be a firing sequence from tr to tr'. A branch $(t_b, tr_b) \in Branch(tr')$ denotes an *opened branch* in σ if $\forall (t_a, tr_a) \in Branch(tr), v_0(tr_a) \neq v_0(tr_b)$.

In the same way, if a branch is not permanent and occurs in the initial marking then it has been "closed" by a final transition.

Definition 4 (Closed branch). Let tr, tr' be two extended markings and σ be a firing sequence from tr to tr'. A branch $(t_a, tr_a) \in Branch(tr)$ denotes an *closed branch* in σ if $\forall (t_b, tr_b) \in Branch(tr'), v_0(tr_a) \neq v_0(tr_b)$.

At last, some branches may appear in an intermediate marking and disappear before the final marking.

Definition 5 (Transient branch). Let tr, tr' be two extended markings and σ be a firing sequence from tr to tr'. A branch (t_c, tr_c) of an extended marking tr'' visited by σ is a *transient branch* if
$\forall (t_a, tr_a) \in Branch(tr) \cup Branch(tr'), v_0(tr_a) \neq v_0(tr_c)$.

We want to split the (possibly empty) set of sequences which leads from the initial marking to the final marking depending on the behavior of branches of the initial and final marking. In the decision procedure, we will try to find successively a sequence in each of this subset. Let us note that the number of these subsets is finite. So an admissible combination of branches of the initial marking and final marking binds the permanent branches and the remainding branches are either closed (if they are in the initial marking) or opened (if they are in the final marking).

Definition 6 (Admissible combination of branches). Let tr and tr' be two extended markings, an admissible combination of branches $Cb(tr, tr')$ is defined by $Cb(tr, tr') = \{R_t\}_{t \in T_{ab}}$such that

- $\forall t \in T_{ab}, R_t \subseteq (branch(tr,t) \cup \perp) \times (branch(tr',t) \cup \perp) \wedge$
- $\forall t \in T_{ab}, \forall b_i \in branch(tr,t), |R_t(b_i, \bullet)| = 1 \wedge$
- $\forall t \in T_{ab}, \forall b_j \in branch(tr',t), |R_t(\bullet, b_j)| = 1 \wedge$
- $\forall t \in T_{ab}, |R_t(\perp, \perp)| = 0$

The meaning of an admissible combination $Cb(tr, tr')$ is the following.

Definition 7 (Combination respect). Lets σ be a sequence from tr to tr', then σ respects $Cb(tr, tr')$ iff :
$\forall t \in T_{ab}, \forall b_i \in branch(tr,t), \forall b_j \in branch(tr',t)$:

- $R_t(b_i, \perp)$ implies σ closes the branch b_i,
- $R_t(\perp, b_j)$ implies σ opens the branch b_j and b_j stays opened,
- $R_t(b_i, b_j)$ implies b_i stays opened during the firing of σ and corresponds to b_j in tr'.

The set of possible branch combinations of two extended markings tr and tr' is denoted $CB(tr, tr')$. It is clear that for any sequence there is one and only one admissible combination respected by the sequence.

3.2 Closability of Abstract Transitions

The first step for tackling the reachability problem is to determine whether the tree generated by the firing of an abstract transition may "close" itself.

Definition 8 (Closable abstract transition). An abstract transition t is *closable* if there exists a firing sequence from $\Omega(t)$ to $\perp$. Such a sequence is called a *closing sequence* of the abstract transition t.

From this definition, it is clear that the branches opened in a firing sequence which can be closed in the same sequence, are those which are composed from a closable abstract transition. Moreover, using the sequence depth, we can characterize some minimal closing sequences.

Definition 9 (Minimal closing sequence). A closing sequence σ of an abstract transition t is said *minimal* if for all closing sequence σ' of t, we have $Depth(\sigma) \leq Depth(\sigma')$.

For a given closable abstract transition t, we denote by $Order(t)$ the depth of its minimal closing sequences. The next lemma exhibits a structural property of some minimal closing sequences.

Lemma 10 (Existence of particular minimal closing sequences). *Let t be an abstract closable transition, there exists a minimal closing sequence σ such that: If σ' denotes the prefix of σ where the last transition (a final one) has been deleted then σ' has no opened branches.*

Proof. Let us take a closing sequence. If there is an opened branch before the last firing, then we can delete the firing which opens this branch and all the subsequent firings in this branch. Indeed, the only node modified by the removing is the root node where all its marking after the opening of the branch are now increased with the input tokens of the abstract transition. Thus the new sequence is also firable. The new sequence has at most the same depth as the previous one. Iterating this removing on all the opened branches, we obtain the required minimal closing sequence. □

Lemma 11 (Branches order). *If there exists an abstract transition $t \in T_{ab}$ such that $Order(t) = n$ (with $n > 1$) then there exists an abstract transition t' such that $Order(t') = n - 1$*

Proof. Let σ be a minimal closing sequence of t fulfilling the condition of lemma 10 and $\{tr_1, \ldots, tr_m\}$ the extended markings of depth n reached by σ. Let $\{t_{i,j}\}$ be the transitions labeling the branches of tr_i.

By the definition of σ, we are ensured that $\forall i, j, Order(t_{i,j}) \leq n-1$. Moreover the condition of the lemma 10 ensures that all the branches are closed before the firing of the last transition (the final one) in the root.

Suppose that $\forall i, j, Order(t_{i,j}) < n - 1$, then we can replace the sequence σ by a sequence σ' where the firings in the branch which follow its creation are replaced by some minimal closing sequence of order $< n - 1$. This sequence σ' has an order $< n$ which contradicts the hypothesis of minimality of σ. □

The Algorithm 3.1 computing the set of closable abstract transitions is based on the Lemma 11. From a given recursive Petri net, it returns the corresponding set of closable abstract transitions.

Proposition 12 (Closable abstract transitions). *The Algorithm 3.1 computes exactly the set of closable abstract transitions of the net N.*

Proof. By definition, an abstract transition of order equal to 1 has an associated minimal closing sequence during which no branch is opened and then no abstract transition is fired. Hence, we have to decide if, for a given abstract transition t, a marking from which a final transition is firable can be reached from $\Omega(t)$. The set S_{fi} represents the semi-linear set of markings from which a final transition is firable. The ordinary net Net is obtained removing all abstract and final transitions. A closable abstract transition of order equal to 1 is determined by

Algorithm 3.1 Closable

```
$TransitionSet$ **Closable**($RPN\ N$)
begin
    $Net = N \setminus \{T_{ab} \cup T_{fi}\}$;
    $Computed = \emptyset$;
    $S_{fi} = \{m \in \mathbb{N}^P \mid \exists t \in T_{fi}, m \geq W^-(\bullet, t)\}$;
    $i = 0$;
    do
        $i = i + 1$;
        $New = \emptyset$;
        forall $t \in T_{ab} \setminus Computed$ do
            if $ElemDecide(Net, \Omega(t), S_{fi})$ then
                $New = New \cup \{t\}$;
                $Order(t) = i$;
            fi
        od
        forall $t \in New$ do
            $t_{eq}$ = an elementary transition such that $W^-(\bullet, t_{eq}) = W^-(\bullet, t) \wedge W^+(\bullet, t_{eq}) = W^+(\bullet, t)$;
            $Net = Net \cup \{t_{eq}\}$;
        od
            $Computed = Computed \cup New$;
    while ($New \neq \emptyset$);
    return $Computed$;
end
```

deciding if there exists a marking of S_{fi} reachable in Net from $\Omega(t)$. In ordinary Petri nets, the reachability of a semi-linear set reduces straightforwardly to the reachability of a finite set of markings. The call $ElemDecide((Net, \Omega(t), S_{fi})$ return $true$ if the ordinary net Net can reach a least a marking of S_{fi} from $\Omega(t)$ and $false$ otherwise.

As seen in the demonstration of the Lemma 11, we know that the closable abstract transitions of order equal to n have an associated minimal closing sequence containing branches of order strictly less than n. Moreover, these branches are transient in the subsequence which precedes the firing of the last transition. Moreover, only the marking of the root is relevant to reach S_{fi}. Then we can mimic the consequence at the root level of behavior of these branches by adding to the net, elementary transitions equivalent to the closable abstract transitions of order strictly less than n. A similar construction is used in Proposition 16 and explained there in more details.

Finally, as the number of abstract transitions is finite and each sucessfull round of the external loop picks at least a new one, the algorithm stops. □

For a given recursive Petri net N, we denote by N_{ord} the ordinary net Net obtained at the termination of the Algorithm 3.1.

3.3 Analysis of Sequences Structure

We enumerate four propositions (one per category of branches) which are useful to reduce the problem of existence of a sequence to the existence of other (and simpler in some sense) sequences.

Proposition 13 (Permanent branch condition). *Let tr and tr' be two extended markings. Let $Cb(tr, tr')$ be an admissible combination of branches and t an abstract transition. There exists a sequence σ from tr to tr' which respects $Cb(tr, tr')$ with a permanent branch $R_t(b_i, b_j)$ iff $\exists \sigma', \sigma''$ such that*

$$(M(v_0(tr)), Branch(tr) \setminus b_i) \xrightarrow{\sigma'} (M(v_0(tr')), Branch(tr') \setminus b_j) \wedge$$
$$b_i \xrightarrow{\sigma''} b_j$$

Proof. The demonstration is essentially based on the semantics of recursive Petri nets. If the branch b_i is permanent then the sequence σ does not contain a firing of an abstract transition opening the branch and any firing of final transition in the root of the branch. Then, all the firings in the branch b_i are independent from the ones outside of the branch.

We construct a sequence σ' by projecting σ on the firings which do not concern b_i and a sequence σ'' by projecting σ on the firings concerning the branch b_i. Due to the fact that the firings inside b_i are independent to the ones outside (from the semantics of recursive Petri nets), if σ is firable then the sequence $\sigma'.\sigma''$ is also firable and leads from tr to tr'.

Moreover, the sequence $\sigma''.\sigma'$ has the same properties.

Finally, it is clear that the existence of firing sequences σ' and σ'' is a sufficient condition for reaching tr' from tr via $\sigma'.\sigma''$. □

This proposition is illustrated in the Fig. 6.

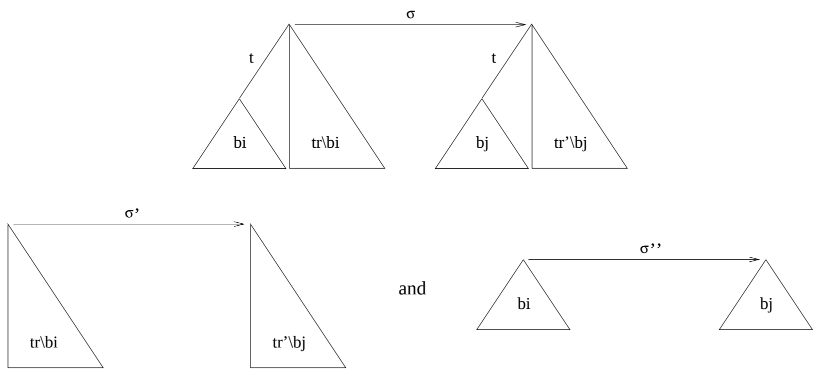

Fig. 6. permanent branch

Proposition 14 (Opened branch condition). *Let tr and tr' be two extended markings. Let $Cb(tr, tr')$ be an admissible combination of branches and t an abstract transition. There exists a sequence σ from tr to tr' which respects $Cb(tr, tr')$ with an opened branch $R_t(\perp, b_j)$ iff $\exists\sigma', \sigma''$ such that*

$$tr \xrightarrow{\sigma'} (M(v_0(tr')) + W^-(\bullet, t), Branch(tr') \setminus \{b_j\}) \wedge$$
$$\Omega(t) \xrightarrow{\sigma''} b_j$$

Proof. t is the abstract transition which opens the branch b_j. Let σ_a be the part of σ preceding the b_j-opening occurrence of t in σ and σ_b the part of σ following this occurrence. We have $\sigma = \sigma_a.t.\sigma_b$.

The branch b_j is a permanent branch of σ_b. Applying the Proposition 13, we construct a sequence σ_{b1} by projecting σ_b on the firings which do not concern the branch b_j and a sequence σ_{b2} by projecting σ_b on the firings concerning b_j.

The sequence $\sigma_a.t.\sigma_{b1}.\sigma_{b2}$ is firable.

Due to the fact that the firing of t only consumes tokens in the root of the extended markings, the sequence $\sigma_a.\sigma_{b1}.t.\sigma_{b2}$ is also firable and leads from tr to tr'.

Moreover, like the firing of t leads to the opening of the branch $(t, \Omega(t))$ and the firings in σ_{b2} only concern b_j, we have $\Omega(t) \xrightarrow{\sigma_{b2}} b_j$.

Since all the firings in σ which do not concern b_j are in $\sigma_a.\sigma_{b1}$, we have $tr \xrightarrow{\sigma_a.\sigma_{b1}} (M(v_0(tr')) + W^-(\bullet, t), Branch(tr') \setminus \{b_j\})$.

Finally, it is clear that the existence of firing sequences σ' and σ'' is a sufficient condition for the reachability via σ'.t.σ'' . □

This proposition is illustrated in the Fig. 7.

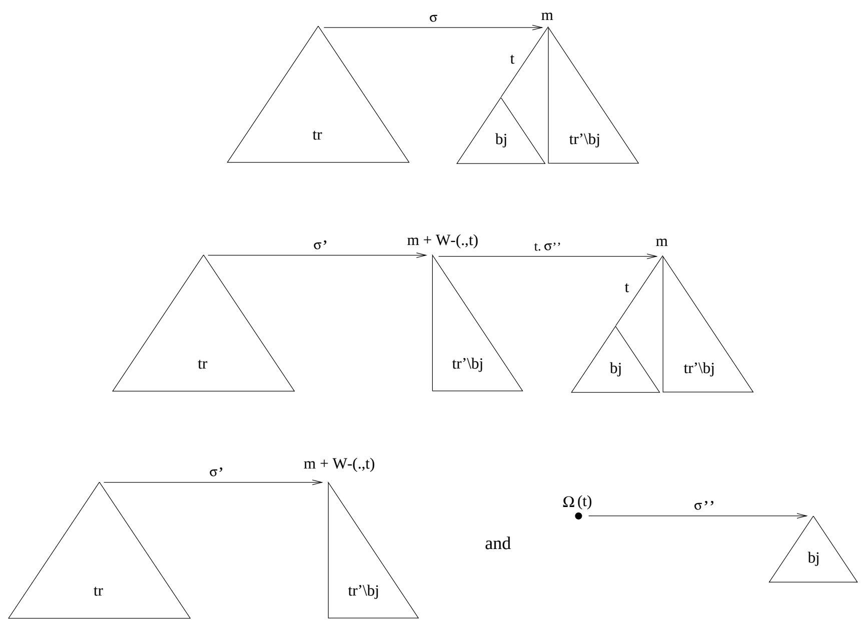

Fig. 7. opened branch

Proposition 15 (Closed branch condition). *Let tr and tr' be two extended markings. Let $Cb(tr, tr')$ be an admissible combination of branches and t an abstract transition. There exists a sequence σ from tr to tr' which respects $Cb(tr, tr')$ with a closed branch $R_t(b_i, \bot)$ iff $\exists \sigma', \sigma''$ such that*

$$b_i \xrightarrow{\sigma'} \bot \wedge$$
$$(M(v_0(tr)) + W^+(\bullet, t), Branch(tr) \setminus \{b_i\}) \xrightarrow{\sigma''} tr'$$

Proof. Lets t_{cl} be the final transition closing the branch b_i. Let σ_a be the part of σ preceding the b_i-closing occurrence of t_{cl} in σ and σ_b the part of σ following this occurrence. We have $\sigma = \sigma_a.t_{cl}.\sigma_b$.

The branch b_i is a permanent branch of σ_a. Applying the Proposition 13, we construct a sequence σ_{a1} by projecting σ_a on the firings which concern the branch b_i and a sequence σ_{a2} by projecting σ_b on the firings not concerning b_i.

The sequence $\sigma_{a1}.\sigma_{a2}.t_{cl}.\sigma_b$ is firable.

Due to the fact that the firing of t_{cl} only produces tokens in the root of the extended markings, the sequence $\sigma_{a1}.t_{cl}.\sigma_{a2}.\sigma_b$ is also firable and leads from tr to tr'.

Because all the firings in σ which do not concern b_i are in $\sigma_{a2}.\sigma_b$, we have $tr \xrightarrow{\sigma_{a1}.t_{cl}} (M(v_0(tr)) + W^+(\bullet, t), Branch(tr) \setminus \{b_i\})$.

Finally, it is clear that the existence of firing sequences σ' and σ'' is a sufficient condition for reachability via $\sigma'.\sigma''$. □

This proposition is illustrated in the Fig. 8.

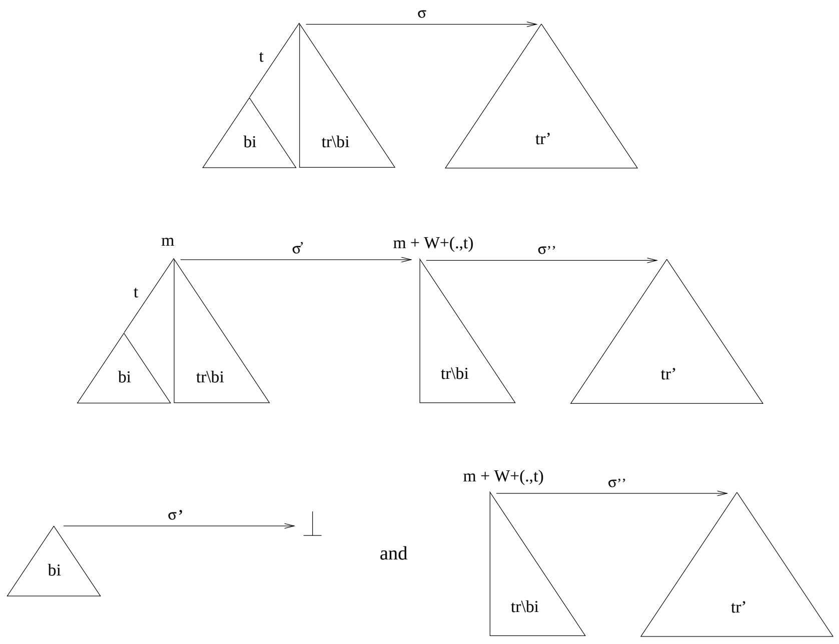

Fig. 8. closed branch

Proposition 16 (Transient branch elimination). *Let tr and tr' be two extended markings composed by only one node. There exists σ a sequence from tr to tr' iff there is a sequence in the ordinary Petri net N_{ord} leading from $M(v_0(tr))$ to $M(v_0(tr'))$.*

Proof. Lets t_{cl} be the final transition closing some transient branch b_i of σ opened by an abstract transition t. Let σ_a be the part of σ preceding the firing of t, σ_b the part of σ enclosed by the opening-firing of t of the branch and the closing-firing of t_{cl} and σ_c be the part of σ following the firing of t_{cl}. We have $\sigma = \sigma_a.t.\sigma_b.t_{cl}.\sigma_c$.

The branch b_i is a permanent branch of σ_b. Applying the Proposition 13, we construct a sequence σ_{b1} by projecting σ_b on the firings which do not concern the branch b_i and a sequence σ_{b2} by projecting σ_b on the firings concerning b_i.

The sequence $\sigma_a.t.\sigma_{b1}.\sigma_{b2}.t_{cl}.\sigma_c$ is firable.

Due to the fact the firing of t only consumes tokens in the root of the extended markings, the sequence $\sigma_a.\sigma_{b1}.t.\sigma_{b2}.t_{cl}.\sigma_c$ is also firable and leads from tr to tr'.

Because the sequence σ_{b2} only concerns the branch b_i opening by t, its firings do not have any effect on the nodes of extended marking reached by $\sigma_a.\sigma_{b1}$. The modifications on these nodes done by the sequence $t.\sigma_{b2}.t_{cl}$ concern only the root node and are the consuming of $W^-(\bullet, t)$ tokens by t and the producing of $W^+(\bullet, t)$ tokens by t_{cl}. So at the root level, the sequence $t.\sigma_{b2}.t_{cl}$ can be simulated by t_{eq} which belongs to N_{ord} as t has a closing sequence. Iterating the substitution for all transient branches gives a sequence in N_{ord}.

Finally, it is clear that a sequence in N_{ord} can be transformed in a sequence for N by substituting the firing of any t_{eq} by the firing of t followed by a closing sequence of t. □

3.4 The Decision Procedure

The Algorithm 3.2 is developed from the previous propositions.

Theorem 17 (Reachability problem). *Let tr and tr' two extended markings of a recursive Petri net $N = \langle P, T, W^-, W^+, \Omega, tr_0\rangle$. tr' is reachable from tr iff $Decide(N, tr, tr')$ returns true.*

Proof. The demonstration is done recursively on $depth(tr) + depth(tr')$.

If $depth(tr) = 0$ then the algorithm returns $true$ only if $depth(tr') = 0$ and $false$ otherwise. This statement is insured by the first test of the algorithm.

We make the hypothesis that the algorithm for $depth(tr) + depth(tr') \leq n$ is correct and demonstrate its correctness for $depth(tr) + depth(tr') = n + 1$.

If $depth(tr) \neq 0 \wedge depth(tr') = 0$, we have to decide if the system is able to fire a final transition at the root level. Or equivalently, if the root of tr is able to reach a state from which a final transition is firable. The set S_{fi} represents this set of markings.

If such a sequence exists, we can adapt the Lemma 10 to it and then, there exists a minimal sequence having no opened branch.

However, a branch opened in tr which can be closed from tr, can increase the number of token in the marking associated to $v_0(tr)$ by producing $W^+(\bullet, t)$ tokens (where t is the abstract transition associated to the branch). Due to the recurrence hypothesis, we can determine these particular branches by recursive calls to the procedure.

Because, adding some tokens to a marking can only increase the number of sequences fired from it, we can arbitrary close these particular branches. The marking m corresponds to this statement. It is clear that the branches opened in tr which can not be closed have no effect on the marking associated to $v_0(tr)$ and then can be ignored.

Finally, because the searched sequence can perform some transient branches and applying the Proposition 16, decide if the system can reach a marking of S_{fi} from m is equivalent to decide if the ordinary net N_{ord} is able to reach a marking of S_{fi} from m. The demonstration of this point is related to the Proposition 16 and the reachability problem for ordinary nets is known to be decidable ([May81,Kos82,Lam88]).

Algorithm 3.2 Decide

```
$Bool$ **Decide**$(N, tr, tr')$
**begin**
    **if** $(tr == \perp)$ **then**
        **return** $tr' == \perp$;
    **if** $(tr' == \perp)$ **then**
        $m = M(v_0(tr))$;
        **forall** $(t, tr_i) \in Branch(tr)$ **do**
            **if** $Decide(N, tr_i, \perp)$ **then**
                $m = m + W^+(\bullet, t)$;
            **fi**
        **od**
        $S_{fi} = \{m \in \mathbb{N}^P \mid \exists t \in T_{fi}, m \geq W_-(\bullet, t)\}$;
        **return** $ElemDecide(N_{ord}, m, S_{fi})$;
    **fi**
    **forall** $Cb(tr, tr') \in CB(tr, tr')$ **do**
    // with $Cb(tr, tr') = \{R_t\}_{t \in T_{ab}}$
        **forall** $R_t((t, tr_i), (t, tr_j))$ **do** // permanent branch
            **if** $\neg Decide(N, tr_i, tr_j)$ **then**
                **goto** $NextCombination$;
            **fi**
        **od**
        **forall** $R_t(\perp, (t, tr_j))$ **do** // opened branch
            **if** $\neg Decide(N, \Omega(t), tr_j)$ **then**
                **goto** $NextCombination$;
            **fi**
        **od**
        **forall** $R_t((t, tr_i), \perp)$ **do** // closed branch
            **if** $\neg Decide(N, tr_i, \perp)$ **then**
                **goto** $NextCombination$;
            **fi**
        **od**
        $m_1 = M(v_0(tr)) + \sum_{R_t(\bullet, \perp)} W^+(\bullet, t)$;
        $m_2 = m_0(tr') + \sum_{R_t(\perp, \bullet)} W^-(\bullet, t)$;
        **if** $ElemDecide(N_{ord}, m_1, m_2)$ **then**
            **return** $true$;
        **fi**
$NextCombination$:
    **od**
    **return** $false$;
**end**
```

If $depth(tr) \neq 0 \wedge depth(tr') \neq 0$ and because the number of combinations is finite, decide if there exists a firing sequence in N leading from tr to tr' is equivalent to decide if there exist a firing sequence σ in N leading from tr to tr' and there exists an admissible combination $Cb(tr, tr')$ such that σ respects $Cb(tr, tr')$.

The main loop of the algorithm corresponds to this statement.

Then, to decide if there exists a sequence in N is equivalent to decide if there exists one showing some permanent, opened, closed and transient branches. The considered admissible combination Cb determines the permanent, opened and closed branches.

The three internal loops correspond to the treatment of these kinds of branch.

The demonstration of there correctness is directly related to the Propositions 13, 14 and 15 and to the induction hypothesis.

When all the permanent, opened and closed branches have been treated, the decision of the reachability can be restricted to a decision in an ordinary net. If we consider that all the closed branches are effectivly closed, we have to reach the state from which all the opened branches can be opened. The markings m_1 and m_2 correspond to these two particular points of the sequence. Because, the sequence can perform some transient branches, the decision must be done for the ordinary net N_{ord} (Proposition 16). $\square$

4 Conclusion

In this work, we have studied theoretical features of recursive Petri nets. We have first shown that they are a strict extension of the model of Petri nets as they are able to recognize the palindrom language and that the languages of RPNs strictly include the union of Petri net and Context Free languages. Moroveover, we have illustrated their modelling capability by giving a simple method to model faults in a system whereas a similar modelling is not known to be possible for ordinary Petri nets.

Then, we have proven that the reachability is decidable for RPNs and we have given an algorithm which reduces the problem to some (quite numerous !) applications of the decision procedure for the ordinary Petri nets.

We plan to extend our studies in two different ways. On the one hand we want to add new features for recursive Petri nets and examine whether the reachability problem remains decidable. We are mainly interested to introduce some context when a thread is initiated (e.g. the starting marking could depend from the depth in the tree). On the other hand, we would study some more complex properties (like home state or properties specified by a temporal formula). Since the redaction of this paper, new results on RPNs have been stated. In particular, we have proven that RPN languages are recursive [HP99b] and that any Turing machine can be simulated by synchronizing a RPN with a finite automaton [HP99a]. This last result has multiple consequences. One of them being that the emptiness of the intersection of a regular language and a RPN language is undecidable. Hence, it leaves little hope to general model checking as done in [Esp97] for Petri nets. However, this negative result does not preclude checking of particular properties (such as special kinds of fairness [Yen92]) and we are working in such a direction.

At last, it would be interesting to combine this model with usual characteristics (such like the colours) in order to increase its application area.

References

[DFS98] C. Dufourd, A. Finkel, and P. Schnoebelen. Reset nets between decidability and undecidability. *(ICALP'98) L.N.C.S*, 1443:103–115, July 1998.

[Esp97] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[HP99a] S. Haddad and D. Poitrenaud. On the expressive power of recursive Petri net. Submitted to the 12th International Symposium on Fundamentals of Computation Theory, FCT'99, August 1999.

[HP99b] S. Haddad and D. Poitrenaud. Recursive Petri net languages are recursive. Submitted to the 10th International Conference on Concurrency Theory, CONCUR'99, August 1999.

[Jan79] M. Jantzen. On the hierarchy of Petri net languages. *RAIRO*, 13(1):19–30, 1979.

[Kos82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc. 14th Annual Symposium on Theory of Computing*, pages 267–281, 1982.

[Lam88] J. L. Lambert. Some consequences of the decidability of the reachability problem for Petri nets. In *Advances in Petri Nets 1988*, volume 340 of *LNCS*, pages 266–282, Berlin, Germany, June 1988. Springer Verlag.

[May81] E.W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. 13th Annual Symposium on Theory of Computing*, pages 238–246, 1981.

[Rei95] K. Reinhardt. Reachability in Petri nets with inhibitor arcs. Unpublished manuscript. See www-fs.informatik.uni-tuebingen.de/ reinhard, 1995.

[SH95] A. El Fallah Seghrouchni and S. Haddad. A formal model for coordinating plans in multiagents systems. In *Proceedings of Intelligent Agents Workshop*, Oxford United Kingdom, November 1995. Augusta Technology Ltd, Brooks University.

[SH96] A. El Fallah Seghrouchni and S. Haddad. A recursive model for distributed planning. In *Second International Conference on Multi-Agent Systems*, Kyoto, Japon, December 1996.

[Val78a] R. Valk. On the computational power of extended Petri nets. *(MFCS'78) L.N.C.S*, 64:526–535, September 1978.

[Val78b] R. Valk. Self-modifying nets, a natural extension of Petri nets. *(ICALP'78) L.N.C.S*, 62:464–476, July 1978.

[Yen92] H-C. Yen. A unified approach for deciding the existence of certain Petri net paths. *Information and Computation*, 96:119–137, 1992.