Acta Informatica manuscript No.
(will be inserted by the editor)

Recursive Petri Nets

Theory and Application to Discrete Event Systems

Serge Haddad[1], **Denis Poitrenaud**[2]

[1] LAMSADE - UMR 7024, Université Paris-Dauphine
Place du Maréchal De Lattre de Tassigny, 75775 Paris cedex 16
e-mail: Serge.Haddad@lamsade.dauphine.fr
[2] LIP6 - UMR 7606, Université Paris VI, Jussieu
4, Place Jussieu, 75252 Paris cedex 05
e-mail: Denis.Poitrenaud@lip6.fr
Tel: (33) 1 44 27 71 04
Fax: (33) 1 44 27 87 71

Received: date / Revised version: date

Abstract. In order to design and analyse complex systems, modelers need formal models with two contradictory requirements: a high expressivity and the decidability of behavioural property checking. Here we present and develop the theory of such a model, the recursive Petri nets. First, we show that the mechanisms supported by recursive Petri nets enable to model patterns of discrete event systems related to the dynamic structure of processes. Furthermore, we prove that these patterns cannot be modelled by ordinary Petri nets. Then we study the decidability of some problems: reachability, finiteness and bisimulation. At last, we develop the concept of linear invariants for this kind of nets and we design efficient computations specifically tailored to take advantage of their structure.

Key words. Petri nets – Expressivity – Reachability problem – Bisimulation – Flows computation

1 Introduction

With the increasing complexity of systems, the use of formal models is a requirement for their design and analysis. However, the modeler is faced with a dilemma: he looks for a highly expressive model while keeping decidable the checking of significant properties. For instance, the inability of automata to represent infinite state systems and the undecidability of termination of systems equivalent to Turing machine exclude them for being practical specification models.

A closer look at the standard patterns of dynamic systems highlights two modelling needs:

1. A model must handle the concurrent execution of parallel sequential processes.
2. It must also manage the dynamical creation of objects (e.g. resources or processes).

We now discuss the advantages and drawbacks of the two most frequently used models: Petri nets (PNs) and process algebras (PA). Whereas modelling concurrent activities is rather straightforward with Petri nets, the management of dynamic objects is limited. Indeed, due to the static structure of the PN, there is no way to keep trace of a synchronisation between two dynamically created processes. With coloured Petri nets [13], such a synchronisation is possible; however, in order to preserve decidability of standard problems, the set of colours (i.e. the number of processes) must be kept finite. From the point of view of verification, the checking of standard properties of PNs and specific properties expressed by event-based linear time logic are decidable problems (see [5] for a survey). Like PNs, process algebra are appropriate for the modelling of concurrency. Furthermore, via recursion and parallel composition, they allow synchronisation between dynamically created objects. However, including these two operators leads to a Turing machine equivalent model.

In this paper, we describe a model which we feel is appropriate with respect to both the expressivity and the analysis capabilities: the recursive Petri nets (RPNs) first introduced in [24].

Previous results: We have already defined two versions of RPNs. In the full model, threads which play the token game of a Petri net, can be dynamically created and concurrently executed. In [7,8], we have studied the expressivity of this model and the checking of some fundamental properties. In particular, we have shown that the RPN model is strictly more expressive than the union of Petri nets and context-free grammars w.r.t. the language point of view. Moreover, we have demonstrated that the reachability problem and some related ones remain decidable for RPNs.

Beside these positive results, we have also shown a negative one in [6]. Whereas the event-based linear time model checking is decidable for Petri nets, it becomes undecidable for RPNs. Thus, in [9], we have introduced a restricted model called sequential RPN (SRPN) for which this problem remains decidable and whose languages family still strictly includes the union of the Petri nets and context-free languages. Roughly speaking, in an SRPN the executions of threads are nested: only the last created one executes until its termination or the creation of a new thread.

Contributions: The first contribution of this paper is the introduction of an extended version of RPNs including new mechanisms. An RPN has the same structure as an ordinary Petri net except that the transitions are partitioned into two categories : elementary and abstract transitions. The semantics of such a net may be informally explained as follows. In an RPN, there is a dynamical tree of threads (denoting the fatherhood relation) where each thread plays its own token game. A step of an RPN is thus a step of one of its threads. If the thread fires an abstract transition, it consumes the input tokens of the transition and generates a new child which begins its own token game with some starting marking depending on the current state of its father. If the thread reaches a marking belonging to a set of final ones, 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. The produced tokens depend on the final marking reached by the thread. In the particular case of the root thread, one obtains an empty tree. If the thread fires an elementary transition, then it updates its current marking using the Petri net firing rule, i.e. it consumes the tokens required by the input arcs and produces the ones specified by the output arcs. Additionally, the firing may abort threads initiated by a previous firing of the current thread.

With respect to the model presented in [8], the current model includes the following additional features: the place capacities, the test arcs, the parametrised initiation and termination of threads and the interrupt capability. We show that the reachability problem remains decidable for this extended version of RPNs. We also prove that the boundedness and the finiteness problems are still decidable. At last, we show how to define and compute some linear invariants which capture relations between a thread and its descendants.

The second contribution of the paper deals with the expressivity of SRPNs. The modelling capabilities of SRPNs can be illustrated in different ways. In this paper, we present the modelling of faults within a system and we demonstrate that an equivalent modelling by an ordinary Petri net is impossible. SRPNs enable also to easily model multi-level executions (e.g. interrupts).

The last contribution is related to bisimulation in the framework of SRPNs. While bisimulation of Petri nets is undecidable, one can check the bisimulation between a Petri net and a finite automaton. We demonstrate that for a subclass of SRPN called restricted SRPN (RSRPN), this problem is still decidable. We emphasise that this restricted model is strictly more expressive than the union of Petri nets and context-free grammars.

Related work: We now relate this model to similar ones. Since the introduction of Petri nets, theoretical works have been developed in order to study the impact of extensions of Petri nets on the analysis capability. 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 (see the unpublished manuscript “Reachability in Petri Nets with Inhibitor arcs” by K. Reinhardt reachable at http://www-fs.informatik.uni-tuebingen.de/~reinhard). 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 [26, 27]. Introducing restrictions on self-modifying nets enables to decide some properties [3] (boundedness, coverability, termination, ...) but the reachability remains undecidable. Moreover, these extensions do not offer a practical way to model the dynamic creation of objects.

In order to tackle this problem, A. Kiehn has introduced a model called net systems [14]. Net systems are a set of Petri nets with special transitions whose firing starts a new token game of one of these nets. A *call* to a Petri net, triggered by such a firing, may return if this net reaches a final marking. All the nets are required to be safe and the constraints associated with the final marking ensure that a net may not return if it has pending calls. It is straightforward to simulate a net system by an RPN. Moreover as the languages of Petri nets are not included in the languages of net systems, the family of net system languages is strictly included in the family of RPN languages.

Another attempt for the introduction of dynamic capability in Petri net is the object Petri nets of R. Valk [28]. In this model, tokens are themselves Petri nets and the creation of a new process is achieved by the production of a new token net. The hierarchy has a limited depth (two levels) and the synchronisation mechanism can operate only between the upper level and one of the subprocess. It has been demonstrated that reachability is an undecidable problem [15]. A similar model called nested Petri nets has been introduced in [18] presenting the following features: the depth of the hierarchy is unbounded and vertical and horizontal synchronizations can operate.

Here again, it has been proved that the reachability and boundedness are undecidable even if termination and other standard properties remain decidable.

Process Algebra Nets (PANs), introduced by R. Mayr [20], are a model of process algebra including the sequential composition operator as well as the parallel one. The left term of any rule of a PAN may use only the parallel composition of variables whereas the right side is a general term. This model includes Petri nets and context-free grammars. We have proved [8] that RPNs also include PANs. Whereas we do not know whether the inclusion of the PAN languages by the RPN ones is strict, we emphasise that the main difference between RPNs and this model is the ability to prune subtrees from the extended marking. For instance, this mechanism is indispensable for the modelling of plans in multi-agents systems [24]. Moreover, PANs as well as Process Rewrite Systems [21] (a more expressive model) cannot represent a transition system with an infinite in-degree.

Recursive-Parallel Programming Scheme (RPPS) [16] also offers the notion of hierarchical state. Like for RPNs, the synchronisation mechanism is restricted to processes linked by the fatherhood relation. Several interesting problems (e.g. reachability) are decidable for this model. However, it has been shown that the family of languages of Petri nets is not included in the one of RPPS.

Organisation of the paper: We have chosen to introduce first the least expressive model. This progressive presentation will help the reader to get an intuitive view of the model capabilities. In the next section, we define sequential recursive Petri nets. Then, we illustrate its expressive power with the modelling of Discrete Event Systems (DES) patterns. Afterwards, we tackle the bisimulation problem between an SRPN and a finite automaton. The third section is dedicated to the full RPN model: we study its expressivity and we establish the decidability of the reachability and related problems. We conclude this section by defining linear invariants and designing algorithms for their computations.

2 Sequential Recursive Petri Nets

2.1 Definitions

Preliminaries: As our model relies heavily on semilinear sets, we briefly introduce their definition and properties. A semilinear set is a subset of $\mathbb{N}^d$ defined as a finite union of linear sets (where

$\mathbb{N} = \{0, 1, \ldots\}$ is the set of natural integers and $d \in \mathbb{N} \setminus \{0\}$). A linear set L is defined by a vector m_0 and a finite set of vectors $\{m_1, \ldots, m_k\}$ such that $L = \{m \mid \exists(\lambda_1, \ldots, \lambda_k) \in \mathbb{N}^k,\ m = m_0 + \sum_{i=1,\ldots,k} \lambda_i \cdot m_i\}$. An *effective representation* is any representation which can be reduced (by an algorithm) to this standard representation. For instance, any system of linear (in)equalities on the coordinates of vectors is an effective representation. Given an effective representation of semilinear sets, the following operations are computable: union, intersection, projection and complementation. Furthermore, the membership problem is decidable (see [4] for details).

In our proofs, we implicitly use the effectiveness of the representation, the above operations and the membership test.

We now introduce a particular kind of semilinear sets. *Restricted* semilinear sets are expressed by a boolean combination of inequalities between a **positive** weighting of the coordinates of vectors and a natural integer.

Definition 1. *A boolean combination ϕ of inequalities w.r.t. to a finite set of variables $\{x_i\}_{1 \leq i \leq d}$ is recursively defined by:*

- *either $\phi =_{def} \Sigma_{1 \leq i \leq d}\, a_i \cdot x_i \leq k$ where $\{a_i\}_{1 \leq i \leq d}$ and k belong to $\mathbb{N}$,*
- *or $\phi =_{def} \phi_1 \wedge \phi_2$ or $\phi =_{def} \phi_1 \vee \phi_2$ or $\phi =_{def} \neg\phi_1$ with ϕ_1, ϕ_2 two boolean combinations of inequalities.*

A restricted *semilinear set E of $\mathbb{N}^d$ is defined by a boolean combination of inequalities ϕ by $E = \{m \mid \phi(\{x_i \leftarrow m(i)\}_{1 \leq i \leq d})$ is true$\}$ where $\phi(\{x_i \leftarrow m(i)\}_{1 \leq i \leq d})$ is the boolean value obtained by setting every variable x_i to the constant $m(i)$.*

Note that a restricted semilinear set is a semilinear set (see [23]) whereas the converse is not true: $x_1 - x_2 = 0$ defines a semilinear set which is not a restricted one.

As an ordinary Petri net, a sequential recursive Petri net has *places* and *transitions*. The transitions are split into two categories: *elementary* and *abstract transitions*.

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 an SRPN there is a stack of threads (each one with its current marking) where the only active thread is on top of the stack. A *step* of an SRPN is thus a step of this thread. The enabling rule of the transitions is specified by the *backward incidence matrix*.

When a thread fires an elementary transition, it consumes the tokens specified by the backward incidence matrix and produces tokens defined by the *forward incidence matrix* (as in ordinary Petri nets).

When a thread fires an abstract transition, it consumes the tokens specified by the backward incidence matrix and creates a new thread (called its son) put on top of the stack which consequently becomes the active one. Such a thread begins its token game with a *starting marking* which depends on the fired abstract transition.

A family of effective representations of semilinear sets of *final markings* is defined in order to describe the termination of the threads. This family is indexed by a finite set whose items are called *termination indexes.* When a thread reaches a final marking, it terminates its token game (i.e. is popped out of the stack). Then it produces in the token game of its father (the new top of the stack) and for the abstract transition which gave birth to him, the tokens specified by the forward incidence matrix. Unlike in the case of ordinary Petri net, this matrix depends also on the termination index of the semilinear set which the final marking belongs to. Such a firing is called a *cut step.* When a cut step occurs in a stack reduced to a single thread, one obtains an empty stack.

The next definitions formalise the model of SRPN and its associated states called *extended markings.*

Definition 2 (Sequential Recursive Petri nets). *A* sequential recursive Petri net *is defined by a tuple $N = \langle P, T, I, W^-, W^+, \varOmega, \varUpsilon \rangle$ where*

- *P is a finite set of places,*
- *T is a finite set of transitions such that $P \cap T = \emptyset$,*
- *A transition of T can be either elementary or abstract. The sets of elementary and abstract transitions are respectively denoted by T_{el} and T_{ab},*
- *I is a finite set of indexes,*
- *W^- is the pre function defined from $P \times T$ to $\mathbb{N}$,*
- *W^+ is the post function defined from $P \times [T_{el} \cup (T_{ab} \times I)]$ to $\mathbb{N}$,*
- *$\varOmega$ is a labelling function from T_{ab} to $\mathbb{N}^P$ which associates with each abstract transition an ordinary marking called the starting marking of t,*
- *$\varUpsilon$ is a family indexed by I of $\underline{\text{effective representations}}$ of semilinear sets of final markings.*

In the sequel, we reason about two kinds of markings: *extended markings* and *ordinary markings.* The former ones are states of SRPN while the latter ones, as in Petri nets, are mappings from P to $\mathbb{N}$

(otherwise stated vectors in $\mathbb{N}^P$). When no confusion is possible, we simply call them markings. We introduce a useful notation for such markings: let $m \in \mathbb{N}^P$ and a set $P' \subseteq P$, the *submarking* $m_{|P'} \in \mathbb{N}^{P'}$ is the restriction of m to P' (i.e. $\forall p \in P', m_{|P'}(p) = m(p)$).

Definition 3 (Extended marking). *An* extended marking tr *of a sequential recursive Petri net* $N = \langle P, T, I, W^-, W^+, \Omega, \Upsilon \rangle$ *is defined by:*

- $d_{tr} \in \mathbb{N}$ *the* depth *of the extended marking and* $\{1, \ldots, d_{tr}\}$ *the set of* levels *of* tr*,*
- $m_1^{tr}, \ldots, m_{d_{tr}}^{tr}$ *the ordinary markings associated with each level,*
- $t_1^{tr}, \ldots, t_{d_{tr}-1}^{tr}$ *a family of abstract transitions indexed by all levels except the last one.*

In order to unify the definitions related to SRPNs and RPNs, we view an extended marking of an SRPN as a tree reduced to a single path where nodes are indexed by levels and labelled by ordinary markings and edges are labelled by abstract transitions. The root of this tree corresponds to the bottom of the stack and the single leaf to its top (for instance, see figure 2).

A marked sequential recursive Petri net (N, tr_0) *is an SRPN* N *together with an initial extended marking* tr_0*.*

According to the presentation, the size of the stack corresponding to an extended marking tr is d_{tr} and the ordinary markings associated with the threads of the stack are $m_1^{tr}, \ldots, m_{d_{tr}}^{tr}$. The empty stack ($d_{tr} = 0$) corresponds to the extended marking without thread and is denoted by $\perp$. Since the effect of cut steps depends on the abstract transition which gave birth to a thread, these transitions $(t_1^{tr}, \ldots, t_{d_{tr}-1}^{tr})$ are stored in the extended marking.

An elementary step will denote either a transition firing or a cut step. The semantics of SRPN given by the enabling and the firing of steps is summarised in the following definitions.

Definition 4. *A transition* t *is* enabled *in an extended marking* tr *(denoted by* $tr\xrightarrow{t}$*) iff* $\forall p \in P, m_{d_{tr}}^{tr}(p) \geq W^-(p,t)$ *and a cut step* τ_i *with* $i \in I$ *is* enabled *(denoted by* $tr\xrightarrow{\tau_i}$*) iff* $m_{d_{tr}}^{tr} \in \Upsilon_i$*.*

Definition 5. *The* firing *of an enabled elementary step* s *of an extended marking* tr *leads to the extended marking* tr' *(denoted by* $tr\xrightarrow{s}tr'$*) depending on the type of* s*.*

- $s \in T_{el}$
  - $d_{tr'} = d_{tr}$, $\forall 0 < i < d_{tr}, m_i^{tr'} = m_i^{tr}, t_i^{tr'} = t_i^{tr}$

  - $\forall p \in P, m^{tr'}_{d_{tr}}(p) = m^{tr}_{d_{tr}}(p) - W^{-}(p,s) + W^{+}(p,s)$
- $s \in T_{ab}$
  - $d_{tr'} = d_{tr} + 1\ ,\ \forall 0 < i < d_{tr}, m^{tr'}_{i} = m^{tr}_{i}, t^{tr'}_{i} = t^{tr}_{i}$
  - $\forall p \in P, m^{tr'}_{d_{tr}}(p) = m^{tr}_{d_{tr}}(p) - W^{-}(s,t)$
  - $t^{tr'}_{d_{tr}} = s\ ,\ m^{tr'}_{d_{tr'}} = \Omega(s)$
- $s = \tau_i$
  - $d_{tr'} = d_{tr} - 1\ ,\ \forall 0 < j < d_{tr} - 1, m^{tr'}_{j} = m^{tr}_{j}, t^{tr'}_{j} = t^{tr}_{j}$
  - $d_{tr'} > 0 \Rightarrow \forall p \in P, m^{tr'}_{d_{tr'}}(p) = m^{tr}_{d_{tr'}}(p) + W^{+}(p, t^{tr}_{d_{tr}-1}, i)$

Let $\{tr_i\}_{1 \leq i \leq n}$ be extended markings, $\{s_i\}_{1 \leq i < n}$ be elementary steps. Then $\sigma = tr_1 \cdot s_1 \cdot tr_2 \cdot s_2 \ldots\ s_{n-1} \cdot tr_n$ is a *firing sequence* (denoted by $tr_1 \xrightarrow{\sigma} tr_n$) iff $\forall 1 \leq i < n, tr_i \xrightarrow{s_i} tr_{i+1}$. In the sequel and for sake of simplicity, σ will be often denoted by $\sigma = s_1 \ldots s_{n-1}$. When multiple SRPNs are involved, we denote by $tr_1 \xrightarrow{\sigma}_N tr_n$ a firing sequence σ in an SRPN N. In a marked SRPN (N, tr_0), an extended marking tr is *reachable* iff there exists a firing sequence $tr_0 \xrightarrow{\sigma} tr$.

Definition 6 (Reachability graph). *Let (N, tr_0) be a marked sequential recursive Petri net. Then its* reachability graph *is defined by $\langle S, A, tr_0 \rangle$ where S, the set of nodes, is the set of reachable extended markings of (N, tr_0), $A \subseteq S \times (T \cup \{\tau_i\}_{i \in I}) \times S$ is the set of edges and $tr_0 \in S$ is the initial node. There is an edge $(tr, t, tr') \in A$ iff $tr \xrightarrow{t} tr'$.*

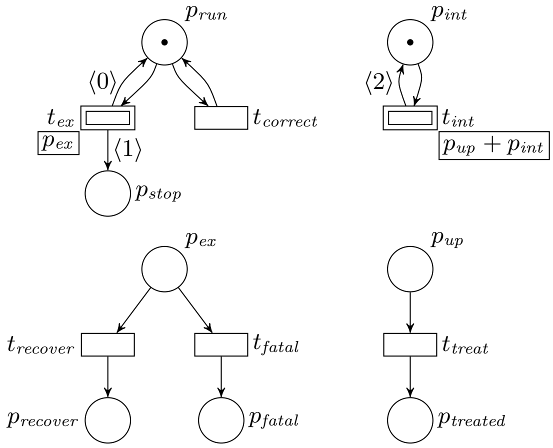
Remarks: When empty, a set of final markings has no effect on the net behaviour, thus in SRPNs we assume that these sets are non empty (contrary to the RPN case to follow). Furthermore, we allow intersection of such sets to be non empty so that the cut steps may be non deterministic. Morevover, $\forall i \in I, \tau_i$ does not belong to T.

2.2 Expressivity of SRPNs

This section illustrates both the syntax and the semantics of SRPN with the help of relevant examples. Furthermore, we simultaneously demonstrate the expressive power of the model and its suitability with respect to standard discrete event system patterns [1].

Modelling of interrupts and exceptions: The net of figure 1 illustrates the characteristic features of SRPNs. The specific features of an SRPN are represented graphically as follow:

- an elementary transition by an ordinary transition (i.e. a single border rectangle),



$\Upsilon_0 = \{m \mid m(p_{recover}) = 1\}$
$\Upsilon_1 = \{m \mid m(p_{fatal}) = 1\}$
$\Upsilon_2 = \{m \mid m(p_{treated}) = 1\}$

Fig. 1 an exception and interrupt mechanism

- an abstract transition t by a double border rectangle whose starting marking $\Omega(t)$ is specified in a frame by $\sum_{p \in P} \Omega(t)(p) \cdot p$, a symbolic sum with null terms omitted and $1 \cdot p$ abbreviated to p.

When defined and non null, the items $W^-(p,t)$ and $W^+(p,t)$ induce arcs with their corresponding integer labels. These labels are omitted when the valuation is equal to one. There is an arc from an abstract transition t to a place p if at least one item $W^+(p,t,i)$ is non null. Such an arc is labelled by the symbolic sum $\sum_{i \in I} W^+(p,t,i) \cdot \langle i \rangle$. As usual, when $W^+(p,t,i) = 0$ the term $W^+(p,t,i) \cdot \langle i \rangle$ is omitted and when $W^+(p,t,i) = 1$ this term is abbreviated as $\langle i \rangle$. Moreover, when for all i the values $W^+(p,t,i)$ are equal, the symbolic sum is abbreviated to this common value. We complete the figure with the definitions of the sets $\{\Upsilon_i\}_{i \in I}$. The set I is implicitly given by the enumeration of these sets.

Figure 1 provides a complete description of an SRPN. Note that contrary to ordinary nets, SRPNs are often disconnected since each connected component may be activated by the firing of different abstract transitions. As the initial extended marking is reduced to a single node, we have directly described the ordinary marking associated with this node by putting token in places. We will follow the same convention in the sequel. The left upper part of the figure models the application level of a processor in an abstract way. The cycle p_{run}, $t_{correct}$, p_{run} represents the correct execution of the current in-

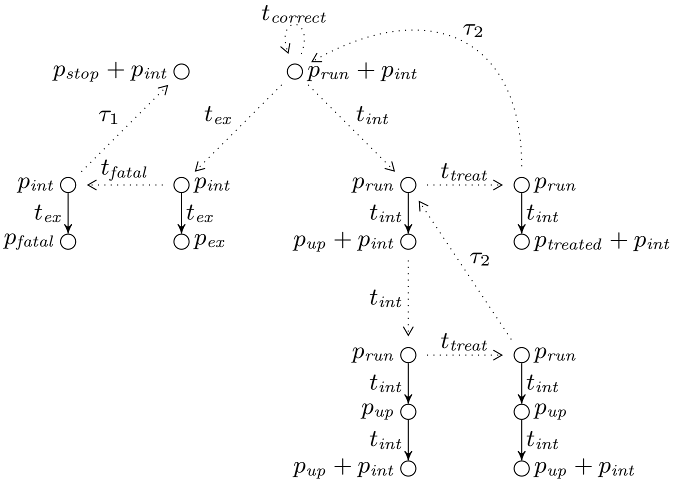


Fig. 2 an extract of the reachability graph of the SRPN of figure 1

struction and the abstract transition t_{ex} models a faulty execution of the instruction yielding a second level with p_{ex} marked. In this level, the system either recovers from the fault ($t_{recover}$) or detects a fatal error (t_{fatal}). The sets $\varUpsilon_0$ and $\varUpsilon_1$ model these two cases. Depending on the fault type, when returning to the first level, the process resumes its activity (place p_{run} marked) or stops it (place p_{stop} marked).

The interrupt modelling outlines the capabilities of the SRPN. When the abstract transition t_{int} is fired, the current execution is interrupted and a second execution level, modelled by a token in p_{up} and p_{int}, is activated. The same construction applies again on this component net making possible a recursive interrupt process. Some variants are conceivable: the number of execution levels could be bounded with additional places, the interrupt could occur during the exception treatment, etc.

Modularity is a natural feature of SRPNs. Indeed, if the system does not include the interrupt mechanism, the modeler simply deletes the right hand side of the figure.

Figure 2 represents an extract of the reachability graph of the SRPN of figure 1. We graphically represent an extended marking as a path whose nodes correspond to levels and are labelled by the associated ordinary markings and whose edges connect level i to $i+1$ and are labelled by t_i^{tr} for $i \in \{1, \ldots, d_{tr} - 1\}$. Extended markings are depicted from top to bottom according to their depth. The dashed arcs denote steps between extended markings.

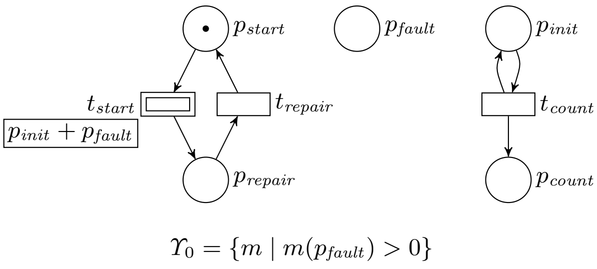


Fig. 3 a basic fault tolerant system

The left part of figure 2 is related to the exception mechanism whereas the right part is devoted to the interrupt occurrences and handling. We just describe a step sequence of the figure. From the initial state, the firing of t_{ex} leads to a two level extended marking. Then the firing of t_{fatal} only changes the ordinary marking of the active node which now belongs to Υ_1. Hence, a cut step τ_1 occurs yielding a single level extended marking where only interrupts may happen.

The example of fig. 1 shows the capability of SRPNs to implicitly keep the context of suspended processes whereas the modelling by Petri nets requires an explicit representation of each context. By use of this capability, we have shown in [8] that SRPNs include the family of algebraic languages. On the other hand, it has been proved that the language of palindromes (a particular algebraic language) cannot be recognised by a labelled Petri net [10]. Thus the language family of SRPNs strictly includes the one of ordinary Petri nets.

Modelling of faults: In order to analyse fault-tolerant systems, the engineer starts from a nominal system and then introduces the faulting behaviour as well as the repairing mechanisms. We limit ourselves to an abstract view of such a system as this pattern may be straightforwardly generalised. The nominal system infinitely executes instructions (elementary transition t_{count}). The marking of place p_{count} represents the number of instruction executions. The complete SRPN is obtained by adding the left part of the figure 3. Its behaviour can be described as follows. There are only two reachable extended markings reduced to a single node: the initial state (tr_{start}) where a token in p_{start} indicates that the system is ready to start and the repairing state (tr_{repair}) where a token in the place p_{repair} indicates that one is repairing the system. Starting from the initial state, the abstract transition t_{start} is fired and the execution of instructions is "played" by the new thread. If this thread dies by a cut step, meaning that a crash occurs, the repairing state is reached. Place p_{fault} represents the possibility of a crash. As p_{fault} is always marked in the correct system and from the very definition of Υ_0, the occurrence of a fault is always possible. We assume that no crash occurs during the repairing stage. With additional places and by modifying Υ_0, we could model more complex fault occurrences (e.g. conditioned by software execution).

The reachable extended markings either consist of a single node or an initial node and its son. However, the number of reachable markings in this latter node is infinite (the place p_{count} is unbounded). In other words, the repairing state can be reached from an infinite number of states which means that *the transition system associated with an SRPN may have some states with an infinite in-degree.* This capability is neither shared by standard Petri nets nor by process algebras. In particular, states in a Petri net reachability graph have an in-degree bounded by $|T|$. Moreover allowing unobservable transitions does not solve the problem. Indeed we demonstrate that the transition system depicted in figure 4 cannot be generated by a standard Petri net with unobservable transitions. First, we formally introduce Petri nets.

Definition 7 (Petri net). *A* labelled marked Petri net *is defined by a tuple* $N = (\langle P, T, W^-, W^+ \rangle, l, m_0)$ *where*

- P *is the finite set of places,*
- T *is the finite set of transitions such that* $P \cap T = \emptyset$*,*
- W^- *and* W^+ *are respectively the pre and post incidence matrices from* $P \times T$ *to* $\mathbb{N}$*,*
- l *is the labelling function from* T *to* $\Sigma \cup \{\lambda\}$ *(with* Σ *an alphabet and* λ *the empty word),*
- $m_0 \in \mathbb{N}^P$ *is the initial marking.*

Definition 8. *Let* $N = (\langle P, T, W^-, W^+ \rangle, l, m_0)$ *be a labelled marked Petri net,* $m \in \mathbb{N}^P$ *be a marking and* $t \in T$ *be a transition. Then*

- t *is enabled in* m *if* $\forall p \in P, m(p) \geq W^-(p,t)$*,*
- *if* t *is enabled in* m *then its firing leads to marking* m' *defined by* $\forall p \in P, m'(p) = m(p) - W^-(p,t) + W^+(p,t)$*, denoted by* $m \xrightarrow{t} m'$*.*

Let σ be a transition sequence, we denote by $m \xrightarrow{\sigma} m'$ the fact that σ leads from m to m'. Such a marking m' is said *reachable* from m. The *reachable set* of a labelled marked Petri net is the set of all markings reachable from the initial marking. As usual, we extend the labelling of transitions to labelling of sequences. We now define the observation graph of a labelled marked Petri net.

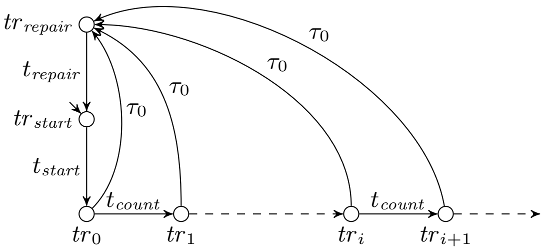


Fig. 4 the (infinite) reachability graph of the SRPN of the figure 3

Definition 9 (Observation graph). *Let $N = (\langle P, T, W^-, W^+\rangle, l, m_0)$ be a labelled marked Petri net. Then the observation graph of this net is defined by $\langle S, A, m_0\rangle$ where $S \subseteq \mathbb{N}^P$, the set of nodes, is the reachability set of the net, $A \subseteq S \times \Sigma \times S$, the set of edges, and $m_0 \in S$, the initial node, is the initial marking of the net. There is an edge $(s, a, s') \in A$ iff there exists a sequence $\sigma \in T^*$ such that $s \xrightarrow{\sigma} s'$ and $l(\sigma) = a \in \Sigma$. Such a sequence is called a witness sequence of the edge.*

The proof of the next theorem is based on the reachability graph depicted in figure 4. tr_{start} is the initial extended marking, tr_i (with $i \in \mathbb{N}$) represents the state of the system where i instructions have been executed and tr_{repair} represents the state where one repairs the system.

Theorem 1 (Infinite in-degree). *Whatever are the labels associated with the SRPN of the figure 3, the graph of the figure 4 cannot be the observation graph of any labelled marked Petri net.*

Proof. Assume that there exists a labelled marked Petri net which generates an observation graph isomorphic to the graph of figure 4. Let us denote m_i (resp. m_{repair}) the marking of this net corresponding to the extended marking tr_i (resp. tr_{repair}). Since the sequence $m_0, m_1, \ldots$ includes only distinct markings, we extract from it a subsequence $m_{\alpha(0)}, m_{\alpha(1)}, \ldots$ composed of strictly increasing markings: $\forall i < j, \forall p \in P, m_{\alpha(i)}(p) \leq m_{\alpha(j)}(p) \wedge \exists p \in P, m_{\alpha(i)}(p) < m_{\alpha(j)}(p)$. Starting from $m_{\alpha(0)}$, there are at least two witness sequences, one leading to m_{repair} (denoted σ) labelled by the cut step τ_0 and the other one leading to $m_{\alpha(0)+1}$ (denoted σ'). These two sequences are also enabled from $m_{\alpha(1)}$. Since $m_{\alpha(0)} \neq m_{\alpha(1)}$, σ does not lead to m_{repair} and thus necessarily leads to $m_{\alpha(1)+1}$. Consequently, the witness sequence from $m_{\alpha(1)}$ to m_{repair} is σ'. Let us examine the witness sequence leading from $m_{\alpha(2)}$ to the initial marking m_0. As previously argued, it cannot be σ thus it is σ'. But this sequence leads also from $m_{\alpha(1)}$ to m_{repair}. Since $m_{\alpha(1)} \neq m_{\alpha(2)}$, there is a contradiction. □

Stated otherwise, the modelling of crash for a system with an infinite number of states is impossible with Petri nets. In the restricted case where the system has a finite number of reachable states, it is theoretically possible to model it with a standard Petri net. However, the modelling of a crash requires a number of transitions proportional to the number of reachable states leading to an intricate net. The SRPN design proposed here does not depend on this number and leads to a compact modelling.

2.3 Analysis of SRPNs

Among the numerous approaches to the verification of systems, a typical one consists in first designing a specification model and an implementing one; and then checking whether the two models are bisimilar. Furthermore, as the specification model is often abstract, it can be modelled by a finite automaton. Consequently, this approach raises the problem of bisimulation between a finite automaton and a general transition system.

Whereas checking the bisimulation of two Petri nets is undecidable [11], checking the bisimulation of a Petri net and a finite automaton becomes decidable [12]. In this subsection, we investigate the generalisation of the latter result to a slightly restricted version of SRPNs.

We will call a *restricted sequential recursive Petri net* (RSRPN), an SRPN whose sets of final markings are restricted semilinear sets. The expressive power of these RSRPN is still high since the proof that the languages family of the SRPN strictly includes the union of the Petri nets and context-free languages is also valid for RSRPN (see [7]). From a practical point of view, let us notice that the two previous examples of SRPNs are restricted ones and more generally, that modelling termination conditions of subsystems very often yields restricted semilinear sets.

The definitions related to the bisimulation are recalled in the appendix A. Note that restricting the SRPN definition is only required for lemma 2.

Lemma 1. *Let (N, tr_0) be a marked SRPN, tr be a reachable extended marking of (N, tr_0) and n be a positive integer. Then there is a reachable extended marking tr' of (N, tr_0) such that $(N, tr) \sim_n (N, tr')$ and $d_{tr'} \leq d_{tr_0} + |T_{ab}| + n$.*

Proof. If $d_{tr} \leq d_{tr_0} + |T_{ab}| + n$, we take $tr' = tr$.
Thus we assume that $d_{tr} > d_{tr_0} + |T_{ab}| + n$. Consequently, the edges corresponding to $t^{tr}_{d_{tr_0}}, \ldots, t^{tr}_{d_{tr_0}+|T_{ab}|}$ have been created by the firing sequence that leads to tr, say σ. Among them, there are at least two occurrences of the same transition that we will denote $t = t^{tr}_i = t^{tr}_j$ with $i < j$.
We can express the firing sequence σ like $\sigma = \sigma_0 \cdot t \cdot \sigma_1 \cdot t \cdot \sigma_2$ with the two occurrences of t corresponding to the production of the labels t^{tr}_i and t^{tr}_j. Now it is straightforward that $\sigma = \sigma_0 \cdot t \cdot \sigma_2$ is also a firing sequence leading to an extended marking tr' with depth strictly less than the one of tr.
As $j \leq d_{tr_0} + |T_{ab}| < d_{tr} - n$, the suffixes of length n (counted as the number of nodes) of the paths corresponding to the extended markings tr and tr' have identical labels (markings and transitions). Thus due to the semantics of an SRPN, the behaviours of the marked nets (N, tr) and (N, tr') are identical up to n firings of steps, which implies that $(N, tr) \sim_n (N, tr')$. If the depth of tr' fulfills $d_{tr'} \leq d_{tr_0} + |T_{ab}| + n$, we are done. Otherwise we iterate this process until fulfillment. □

We introduce some notations related to an RSRPN N and to ordinary markings:
• w_N is the maximum over the set of valuations of the arcs for input places of transitions, i.e. w_N is the least integer such that $\forall p \in P, \forall t \in T, W^-(p,t) \leq w_N$.
• k_N is the maximum over the integers $(k, k', \ldots)$ occurring in the right sides of the inequalities defining the sets of final markings (see definition 1).
• Let m and m' be ordinary markings and H be a positive integer. Then $m \approx_H m'$ iff $\forall p \in P$, $m(p) = m'(p) \vee (m(p) \geq H \wedge m'(p) \geq H)$.

The next lemma establishes a sufficient condition for equivalence of two extended markings w.r.t. $\sim_n$. This condition relies on two observations. First, only the n^{th} last levels of an extended marking determine the firing sequences of length less than or equal to n. Second, if the marking of a place exceeds some bound depending on n then this place will not disable any firing in such a sequence. More precisely, this lower bound can be set to $n \cdot w_N + k_N + 1$.

Lemma 2. *Let N be an RSRPN, tr and tr' be two extended markings of N and n be a positive integer. Let $H = n \cdot w_N + k_N + 1$. Assume that the following assertions are fulfilled:*

1. $(d_{tr} \geq n \wedge d_{tr'} \geq n) \vee d_{tr} = d_{tr'}$

2. $\forall 0 < i < min(n, d_{tr}), t^{tr}_{d_{tr}-i} = t^{tr'}_{d_{tr'}-i}$
3. $\forall 0 \leq i < min(n, d_{tr}), m^{tr}_{d_{tr}-i} \approx_H m^{tr'}_{d_{tr'}-i}$

Then $(N, tr) \sim_n (N, tr')$*.*

Proof. Again we notice that the behaviour of an SRPN up to n firings depends only on the suffix of length n of the path corresponding to the extended marking. We prove the lemma by induction on n. The base case $n = 0$ is trivial. Now assume that $n > 0$. Let us look at a possible firing in (N, tr). If this firing is an elementary or abstract transition t then t is also enabled in (N, tr') since, for every $p \in P$, the markings $m^{tr}_{d_{tr}}(p)$ and $m^{tr'}_{d_{tr'}}(p)$ are either identical or $m^{tr'}_{d_{tr'}}(p) \geq H$ does not disable t by definition of H.

Firing t in the two marked nets leads to new extended markings which fulfill the assertion for $n - 1$. Indeed, the markings where the firing occurs have for a place p either identical markings or a new lower bound $(n-1) \cdot w_N + k_N + 1$. The other markings are unchanged and still satisfy the assertions.

In case where t is an abstract transition, the transition of the new edge is t and the marking of the new node is $\Omega(t)$ for both new extended markings.

Let us examine the case of a cut step. We claim that an inequality occurring in the definition of Υ is either satisfied by both $m^{tr}_{d_{tr}}$ and $m^{tr'}_{d_{tr'}}$ or not satisfied by both $m^{tr}_{d_{tr}}$ and $m^{tr'}_{d_{tr'}}$.

If this inequality does not involve places with different markings, we are done. Otherwise, let p be such a place i.e. with $m^{tr}_{d_{tr}}(p) \neq m^{tr'}_{d_{tr'}}(p)$ and the corresponding coefficient of the inequality $a_p \neq 0$ (see def. 1). Then these markings are greater than or equal to H and due to the positive coefficients of the inequality, this inequality is not satisfied for both the markings. Hence the evaluation of a boolean combination of inequalities will be identical for the two markings. Thus the cut step is also enabled in (N, tr'). The firing of the cut step leads to two extended markings where the suffixes of length $n - 1$ still satisfy the assertions (indeed the marking of places in the new leaf may only be increased). □

Let H be an integer, an extended marking tr is H-bounded iff $\forall p \in P, \forall 1 \leq i \leq d_{tr}, m^{tr}_i(p) \leq H$. Given tr an H-bounded extended marking, $C_H(tr)$ denotes the set of extended markings such that $tr' \in C_H(tr)$ iff:

- $d_{tr} = d_{tr'}$
- $\forall 1 \leq i < d_{tr}, t^{tr}_i = t^{tr'}_i$

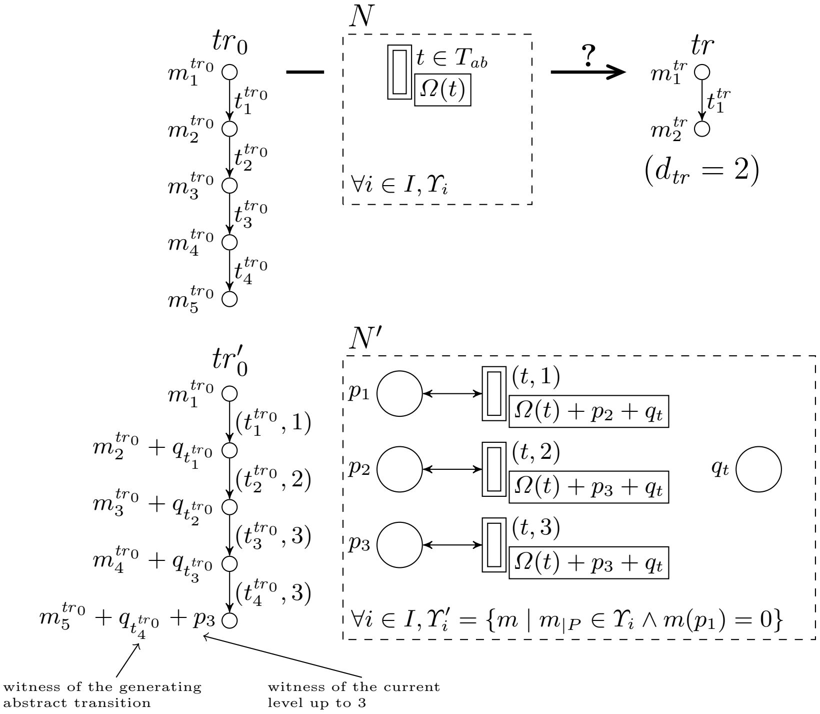


Fig. 5 an illustrative scheme of the construction of lemma 3 (first part)

- $\forall 1 \leq i \leq d_{tr}, m_i^{tr} \approx_H m_i^{tr'}$

Lemma 3. *Let (N, tr_0) be a marked SRPN, H be an integer and tr be a H-bounded extended marking of N. Then one can build a marked SRPN (N'', tr''_0) such that $\perp$ is reachable from (N'', tr''_0) if and only if there exists an extended marking of $C_H(tr)$ reachable from (N, tr_0).*

Proof. We assume that $tr_0 \neq \perp$ since $C_H(\perp) = \{\perp\}$ and, in consequence, we can set $(N'', tr''_0) = (N, tr_0)$.

First, we build an intermediate SRPN (N', tr'_0) obtained by a transformation of (N, tr_0) (see Figure 5). We add two sets of places $\{p_1, p_2, \ldots, p_{d_{tr}+1}\}$ and $\{q_t \,|\, t \in T_{ab}\}$. Any abstract transition t is replaced by $d_{tr} + 1$ copies $(t, 1), (t, 2), \ldots, (t, d_{tr} + 1)$ such that $\forall d \leq d_{tr} + 1, (t, d)$ has p_d as additional input and output place. These additional arcs are the only ones connected to the new places. Moreover $\forall d \leq d_{tr}$, in $\Omega'((t, d))$, p_{d+1} and q_t are marked and in $\Omega'((t, d_{tr} + 1))$, $p_{d_{tr}+1}$ and q_t are marked.

tr'_0 has the same depth as tr_0 and $\forall d < d_{tr_0}$, if $d \leq d_{tr}$ then $t_d^{tr'_0} = (t_d^{tr_0}, d)$ else $t_d^{tr'_0} = (t_d^{tr_0}, d_{tr} + 1)$. The markings of nodes of tr'_0

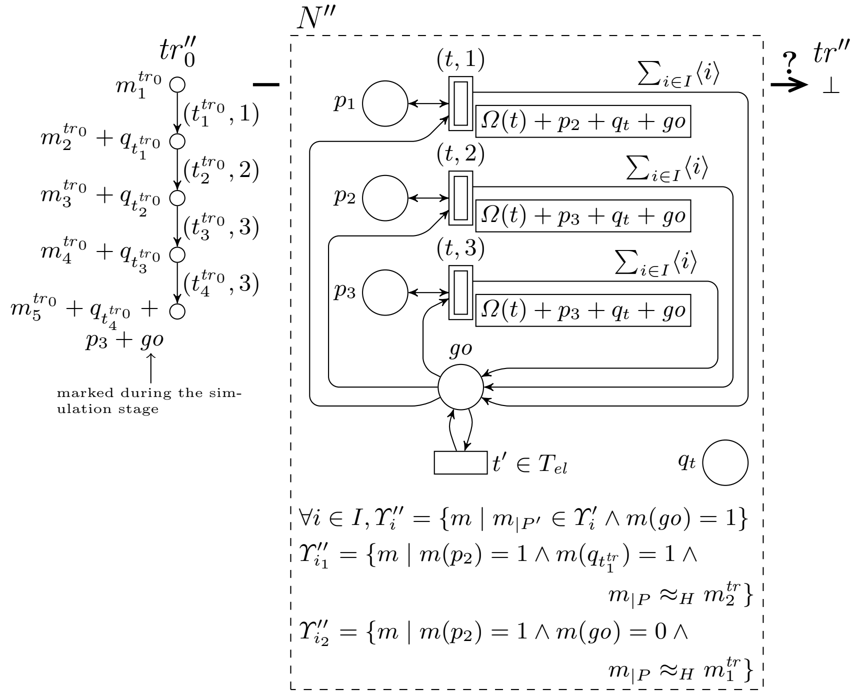

tr''_0
$m_1^{tr_0}$
$(t_1^{tr_0},1)$
$m_2^{tr_0}+q_{t_1^{tr_0}}$
$(t_2^{tr_0},2)$
$m_3^{tr_0}+q_{t_2^{tr_0}}$
$(t_3^{tr_0},3)$
$m_4^{tr_0}+q_{t_3^{tr_0}}$
$(t_4^{tr_0},3)$
$m_5^{tr_0}+q_{t_4^{tr_0}}+$
p_3+go
marked during the simulation stage
N''
$(t,1)$
p_1
$\sum_{i\in I}\langle i\rangle$
$\Omega(t)+p_2+q_t+go$
$(t,2)$
p_2
$\sum_{i\in I}\langle i\rangle$
$\Omega(t)+p_3+q_t+go$
$(t,3)$
p_3
$\sum_{i\in I}\langle i\rangle$
$\Omega(t)+p_3+q_t+go$
go
$t'\in T_{el}$
q_t
$\forall i\in I, \Upsilon''_i=\{m \mid m_{|P'}\in\Upsilon'_i \wedge m(go)=1\}$
$\Upsilon''_{i_1}=\{m \mid m(p_2)=1 \wedge m(q_{t_1^{tr}})=1 \wedge m_{|P}\approx_H m_2^{tr}\}$
$\Upsilon''_{i_2}=\{m \mid m(p_2)=1 \wedge m(go)=0 \wedge m_{|P}\approx_H m_1^{tr}\}$
?
tr''
$\perp$

Fig. 6 an illustrative scheme of the construction of lemma 3 (second part)

are defined as follows:

- $\forall 1 \le d \le d_{tr_0}, \forall p \in P, m_d^{tr'_0}(p) = m_d^{tr_0}(p)$,
- $\forall t \in T_{ab}, m_1^{tr'_0}(q_t) = 0$,
- $\forall 1 < d \le d_{tr_0}, \forall t \in T_{ab}$, if $t_{d-1}^{tr_0} = t$ then $m_d^{tr'_0}(q_t) = 1$ else $m_d^{tr'_0}(q_t) = 0$,
- $\forall 1 \le d \le d_{tr_0}, \forall 1 \le d' \le d_{tr}+1$, if $d' = d = d_{tr_0} \wedge d \le d_{tr}$ then $m_d^{tr'_0}(p_{d'}) = 1$ else if $d' = d_{tr}+1 \wedge d = d_{tr_0} \wedge d > d_{tr}$ then $m_d^{tr'_0}(p_{d'}) = 1$ else $m_d^{tr'_0}(p_{d'}) = 0$.

In the new net, $m \in \Upsilon'_i$ iff the projection on the original places belong to Υ_i and p_1 is unmarked.

Given a firing sequence of (N, tr_0) not leading to $\perp$, one can build a firing sequence of (N', tr'_0) by substituting for the firing of an abstract transition t the firing of (t,d) where $d = min(d', d_{tr}+1)$ and d' denotes the level where the firing has occurred. Moreover, the intermediate extended markings coincide on places of P and in N', all places p_d and q_t are appropriately marked. More precisely, in a marking m

occurring at level $k \leq d_{tr} + 1$, $m(p_k) = 1 \wedge \forall k' \neq k, m(p_{k'}) = 0$ and in a marking m occurring at level $k > d_{tr} + 1$, $m(p_{d_{tr}+1}) = 1 \wedge \forall k' \neq d_{tr} + 1, m(p_{k'}) = 0$. If this level has been created by the firing of the abstract transition t then $m(q_t) = 1 \wedge \forall t' \neq t, m(q_{t'}) = 0$. At the root level, all places q_t are unmarked.

Conversely, given a firing sequence of (N', tr'_0), one can build a firing sequence of (N, tr_0) by substituting for the firing of an abstract transition (t, d) the firing of t. Moreover the intermediate extended markings coincide on places of P. We also note that no firing sequence of (N', tr'_0) leads to $\perp$ since at the root level, p_1 is marked and every Υ_i requires that $m(p_1) = 0$.

Now we make the second transformation leading from (N', tr'_0) to the construction of the marked net (N'', tr''_0) (see Figure 6). We add a new place go. Considering places different from go, tr''_0 is identical to tr'_0, $\forall 1 < d < d_{tr_0}, m_d^{tr''_0}(go) = 0$ and $m_{d_{tr_0}}^{tr''_0}(go) = 1$.

Furthermore, $\forall t \in T''_{ab}, \Omega''(t) = \Omega'(t) + go$. We now partially define the connection between go and the transitions: $\forall t \in T'', W''^-(go, t) = 1$ and $\forall t \in T''_{el}, W''^+(go, t) = 1$. The remaining connections will be defined within the presentation of the new set of indices.

This set is $I'' = I \cup \{i_1, i_2\}$. $\forall i \in I, m \in \Upsilon''_i$ iff $m(go) = 1$ and the projection on the other places belongs to Υ'_i. And $\forall i \in I, \forall t \in T'_{ab}, W''^+(go, t, i) = 1$. Υ''_{i_1} is the set of markings m' s.t.:

- $m'(p_{d_{tr}}) = 1$,
- $d_{tr} > 1 \Rightarrow m'(q_{t^{tr}_{d_{tr}-1}}) = 1$,
- $m^{tr}_{d_{tr}} \approx_H m'_{|P}$.

Thus this cut step is only enabled at the level d_{tr}. Υ''_{i_2} is the set of markings m' s.t. $\exists 1 \leq d < d_{tr}$:

- $m'(p_d) = 1$,
- $d > 1 \Rightarrow m'(q_{t^{tr}_{d-1}}) = 1$,
- $m^{tr}_d \approx_H m'_{|P}$,
- $m'(go) = 0$.

Thus this cut step is only enabled at the levels $1 \leq d < d_{tr}$. These two cut steps do not produce any token: $\forall p \in P'', t \in T''_{ab}, i \in \{i_1, i_2\}, W''^+(p, t, i) = 0$.

Let σ be a firing sequence of (N'', tr''_0) which does not include τ_{i_1} and τ_{i_2} steps. This sequence corresponds to a firing sequence of the original net with the following additional informations: in every visited extended marking, the marking of places p_d witnesses the level where this marking occurs up to level $d_{tr} + 1$, the marking of places

q_t indicates which abstract transition has created the level and the place *go* is marked only at the higher level. Similarly, a firing sequence of the original net (which does not lead to $\perp$) can be simulated in (N'', tr''_0) with the same additional informations.

Since a τ_{i_2} step requires the place *go* to be unmarked it can only be fired if at level d_{tr} there has been a τ_{i_1} step.

Thus the behaviour of this net is the following one: it simulates the original net until it reaches an extended marking tr'' corresponding to an extended marking tr' of the original net with $d_{tr'} = d_{tr}$ such that at this level , tr' fulfills the condition of $C_H(tr)$ related to level d_{tr}, i.e. $t^{tr}_{d_{tr}-1} = t^{tr'}_{d_{tr}-1}$ and $m^{tr}_{d_{tr}} \approx_H m^{tr'}_{d_{tr}}$ which is equivalent to $m^{tr''}_{d_{tr}} \in \Upsilon''_{i_1}$.

Then it performs a cut step τ_{i_1}. The simulation is now stopped (the place *go* is unmarked) and the execution can only produce τ_{i_2} steps until $\perp$ is reached iff at every level k decreasing from $d_{tr} - 1$ to 1 all the intermediate markings and abstract transitions firings of tr' fulfill the conditions of $C_H(tr)$ related to k, i.e. $t^{tr}_{k-1} = t^{tr'}_{k-1}$ (when $k > 1$) and $m^{tr}_k \approx_H m^{tr'}_k$.

When the net stops the simulation, the current extended marking may not correspond to an extended marking of $C_H(tr)$. In this case, the net will reach a deadlock different from $\perp$ since one of the τ_{i_2} step will not be enabled.

Thus $\perp$ is reachable in (N'', tr''_0) iff an extended marking of $C_H(tr)$ is reachable from (N, tr_0). □

In order to state the main result of this section, we introduce transition labels defined by a label mapping l. $l(t)$, the label of transition t, is either a letter of a finite alphabet or the empty word. Cut steps are similarly labelled by l. We note $tr_0 \xrightarrow{l(t)}_N tr_1$ whenever $tr_0 \xrightarrow{t}_N tr_1$.

Theorem 2 (Bisimulation of an RSRPN). *The problem of bisimulation between a labelled marked RSRPN (N, l, tr_0) where no transition or cut step is labelled by the empty word and a finite transition system (LTS, s_0) is decidable.*

Proof. Let n be the number of states of LTS. Due to lemma 6 of the appendix A, it suffices to check whether $(N, l, tr_0) \sim_n (LTS, s_0)$ and whether there exists an extended marking tr_1 reachable from tr_0 s.t. $tr_1 \in Inc^{LTS}_n$.

In order to check $(N, l, tr_0) \sim_n (LTS, s_0)$, we verify that:

1. for every step $tr_0 \xrightarrow{a}_N tr_1$, there exists s_1 s.t. $s_0 \xrightarrow{a}_{LTS} s_1$ and $(N, l, tr_1) \sim_{n-1} (LTS, s_1)$,

2. for every s_1 s.t. $s_0 \xrightarrow{a}_{LTS} s_1$, there exists a step $tr_0 \xrightarrow{a}_N tr_1$ s.t. $(N, l, tr_1) \sim_{n-1} (LTS, s_1)$.

These conditions straightforwardly lead to a recursive procedure where the number of nested calls is bounded by n.

Due to lemma 1, in order to check whether there exists tr_1 reachable from tr_0 s.t. $tr_1 \in Inc_n^{LTS}$ we can restrict the search to extended markings whose depth is bounded by $d_{tr_0} + |T_{ab}| + n$.

Due to lemma 2, each such extended marking tr_1 belongs to some $C_H(tr)$ with $H = n \cdot w_N + k_N + 1$ and tr being H-bounded, whose depth is bounded by $d_{tr_0} + |T_{ab}| + n$ and such that $(N, l, tr) \sim_n (N, l, tr_1)$. More precisely, $d_{tr} = d_{tr_1}, \forall 1 \leq d < d_{tr}, t_d^{tr} = t_d^{tr_1}$ and $\forall 1 \leq d \leq d_{tr}, \forall p \in P, m_d^{tr_1}(p) \leq H \Rightarrow m_d^{tr}(p) = m_d^{tr_1}(p)$ and $m_d^{tr_1}(p) > H \Rightarrow m_d^{tr}(p) = H$.

Let us call TR the set of H-bounded extended markings whose depth is bounded by $d_{tr_0} + |T_{ab}| + n$. This set is finite. For each $tr \in TR$, we check whether an extended marking of $C_H(tr)$ is reachable in (N, tr_0) using the procedure described in lemma 3: we reduce it to a reachability problem for (N'', tr_0'') (decidable by theorem 3 to be shown in section 3.4 for general recursive Petri nets). If some $tr' \in C_H(tr)$ is reachable then using the procedure described in the beginning of the proof, we check for every state $s \in LTS$ whether $(N, l, tr) \sim_n (LTS, s)$. If no s is found for some $tr \in TR$ then $(N, l, tr) \in Inc_n^{LTS}$ and since $(N, l, tr') \sim_n (N, l, tr)$ then $(N, l, tr') \in Inc_n^{LTS}$. So the second condition of lemma 6 is not fulfilled when some $tr' \in C_H(tr)$ is reachable from tr_0. The algorithm 1 describes the procedure associated with this proof. □

Observation. In order to abstract the call mechanism, it would be interesting to hide abstract transitions and cut steps by labelling them with the empty word. Unfortunately, the problem of the bisimulation between a Petri net with some empty labels and a LTS is already undecidable [12]. A partial solution would be to label such items by specific labels (e.g. "call" and "return").

3 Recursive Petri Nets

Discussion. Let us emphasize the applicability and limits of SRPNs. They are appropriate when dealing with real-time systems composed by several static tasks ranked by execution level or with collaborative applications where the interactions are based on synchronous remote procedure calls. However, as soon as the tasks can be dynamically

Algorithm 1: CheckBisimulation

```
input  : a marked RSRPN $(N, tr_0)$ and a finite LTS $(LTS, s_0)$
output: a boolean indicating if $(N, tr_0) \sim (LTS, s_0)$

CheckBisimulation() begin
    //n is the number of states of LTS
    return CheckCond1($tr_0, s_0, n$) and CheckCond2();
end

CheckCond1($tr, s, l$) begin
    if $l == 0$ then return true;
    foreach $tr \xrightarrow{a}_N tr_1$ do
        if not ($\exists s_1$ s.t. $s \xrightarrow{a}_{LTS} s_1$ and CheckCond1($tr_1, s_1, l-1$)) then
            return false;
        end
    end
    foreach $s \xrightarrow{a}_{LTS} s_1$ do
        if not ($\exists tr_1$ s.t. $tr \xrightarrow{a}_N tr_1$ and CheckCond1($tr_1, s_1, l-1$)) then
            return false;
        end
    end
    return true;
end

CheckCond2() begin
    foreach H-bounded $tr'$ whose depth is bounded by $d_{tr} + |T_{ab}| + n$ do
        //using lemma 3
        //and the reachability procedure for RPNs
        if CheckReach($tr_0, C_H(tr')$) then
            found = false;
            foreach state $s'$ of LTS do
                found = found or CheckCond1($tr', s', n$);
            end
            if not found then return false;
        end
    end
    return true;
end
```

created or the calls are asynchronous, the SRPN model is not enough powerful. This motivates the introduction of recursive Petri nets.

3.1 Definitions

SRPNs enlarge Petri nets by introducing a “function call” like mechanism. However, concurrency (the main feature of Petri nets) is confined inside the single active node. In *Recursive Petri Nets* (RPNs), all the nodes are active and therefore concurrency also occurs between node activities. Thus, in an RPN there is a tree of threads (denoting the fatherhood relation) where all the threads play their own token game. A step of an RPN is thus a step of one of its threads. This has an immediate consequence on the cut steps: when a thread performs a cut step the subtree whose root it is pruned.

Furthermore, RPNs include additional mechanisms which enable the modeler to express various kinds of control between threads.

- **Place capacities**: Each place has a specific bound on the number of tokens it can contain. This bound may be infinite meaning that there is no constraint on the place. The set of places with a finite bound is denoted Q.
- **Test arcs**: A new kind of arcs, called test arcs, checks for the exact number of tokens in a place. This place must belong to Q (see the end of this section for a discussion about test arcs).
- **Variable starting markings**: The starting marking of an abstract transition may depend on the current ordinary marking of the thread which fires it. More precisely, this dependence is restricted to the submarking of places in Q.
- **Thread interrupts**: The behaviour of an elementary transition is now twofold and depends on a partial function which associates with it, a set of abstract interrupted transitions and the indexes of the termination. Thus on the one hand, an elementary transition consumes and produces the tokens specified by the backward and forward matrices as an elementary transition of an SRPN. On the other hand, it deletes some subtrees according to the partial function.

The next definitions formalise the syntax and the semantics of RPNs.

Definition 10 (Recursive Petri nets). *A* recursive Petri net *is defined by a tuple* $N = \langle P, \mathcal{B}, T, I, W^-, W^*, W^+, \Omega, \Upsilon, K \rangle$ *where*

- P *is a finite set of places,*
- $\mathcal{B}$ *is the bounding function from* P *to* $\mathbb{N} \cup \{\infty\}$ *inducing the following notations:*
  - $Q = \{p \in P \mid \mathcal{B}(p) \neq \infty\}$ *the subset of bounded places,*
  - $\mathcal{B}[Q] = \{m \in \mathbb{N}^Q \mid \forall q \in Q, m(q) \leq \mathcal{B}(q)\}$*,*
  - $\mathcal{B}[P] = \{m \in \mathbb{N}^P \mid \forall p \in P, m(p) \leq \mathcal{B}(p)\}$*,*
- T *is a finite set of transitions such that* $P \cap T = \emptyset$*,*
- *a transition of* T *can be either elementary or abstract. The sets of elementary and abstract transitions are respectively denoted by* T_{el} *and* T_{ab}*,*

- *I is a finite set of indexes,*
- *W^- is the pre function from $P \times T$ to $\mathbb{N}$,*
- *W^* is the test partial function from $Q \times T$ to $\mathbb{N}$,*
- *W^+ is the post function from $P \times [T_{el} \cup (T_{ab} \times I)]$ to $\mathbb{N}$,*
- *Ω is the starting marking function from $T_{ab} \times \mathcal{B}[Q]$ to $\mathcal{B}[P]$,*
- *Υ is a family indexed by I of effective representations of semilinear sets of final markings,*
- *K is a partial function from $T_{el} \times T_{ab}$ to I such that $\forall t \in T_{ab}$, $(\exists t' \in T_{el}, K(t',t)$ is defined $\Rightarrow$*
$$\forall p \in Q, W^-(p,t) = 0, \forall i \in I, W^+(p,t,i) = 0).$$[1]

Notations and terminology:

- Let t be an abstract transition, $K(t)$ denotes the set of elementary transitions which interrupt t (i.e. $K(t) = \{t' \mid K(t',t) \text{ is defined}\}$). Remark that the index relative to an interrupt is deterministically selected contrary to the index of the cut steps.
- Let t be an elementary transition, $K(t)$ denotes the set of abstract transitions which are interrupted by t (i.e. $K(t) = \{t' \mid K(t,t') \text{ is defined}\}$).
- We denote by $\overline{Q}$ the set $P \setminus Q$.
- We will often focus on the subnet generated by Q. Thus we introduce a useful abbreviation. Let t be a transition and m a submarking on Q then m is *compatible* with t iff $\forall q \in Q, (W^*(q,t)$ is undefined or $m(q) = W^*(q,t))$ and $m(q) \geq W^-(q,t)$.

Definition 11 (Extended marking). *An* extended marking *tr of a recursive net $N = \langle P, \mathcal{B}, T, I, W^-, W^*, W^+, \Omega, \Upsilon, K\rangle$ is a labelled rooted tree directed from the root to the leaves $tr = \langle V, M, E, A\rangle$ where*

- *V is the (possibly empty) finite set of nodes. When it is non empty $v_0 \in V$ denotes the root of the tree,*
- *M is a mapping from V to $\mathbb{N}^P$ associating an ordinary marking with any node and such that $\forall v \in V, \forall p \in P, M(v)(p) \leq \mathcal{B}(p)$,*
- *$E \subseteq V \times V$ is the set of edges,*
- *A is a mapping from E to T_{ab} associating an abstract transition with any edge.*

A marked recursive Petri net *(N, tr_0) is a recursive Petri net N together with an initial extended marking tr_0.*

[1] The requirement that an interruptible abstract transition does not modify the marking of places of Q will be used for the proof of the decidability of the reachability problem (see the first part of the proof of the lemma 5).

When we deal with different extended markings, we will denote the items of an extended marking tr as a function (e.g. $V(tr)$) and more particulary, when tr is non empty, we denote by $v_0(tr)$ the root node. For any node $v \in V$, we denote 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^* stands for the reflexive and transitive closure of E). Moreover, when v is not the root of the tree, we denote by $pred(v)$ its (unique) predecessor. The empty tree is denoted by $\perp$. Any ordinary marking m can be seen as an extended marking, denoted by $\lceil m \rceil$, consisting of a single node.

An *elementary step* of an RPN may be either a firing of a transition or a *cut step* (denoted by τ_i with $i \in I$).

Definition 12. *The* firing *of an elementary transition t from a node v of an extended marking $tr = \langle V, M, E, A\rangle$ leads to the extended marking $tr' = \langle V', M', E', A'\rangle$ (denoted by $tr\xrightarrow{t,v}tr'$) if and only if:*

Let $E'' = \{(v, v') \in E | A((v, v')) \in K(t)\}$ and $V'' = \{v' \in V | (v, v') \in E''\}$,

- $\forall p \in P, M(v)(p) \geq W^-(p, t)$,
- $\forall q \in Q$ *s.t.* $W^*(q, t)$ *is defined,* $M(v)(q) = W^*(q, t)$,
- $V' = V \setminus (\cup_{v' \in V''} Succ(v'))$, $E' = E \cap (V' \times V')$,
- $\forall e \in E', A'(e) = A(e)$, $\forall v' \in V' \setminus \{v\}$, $M'(v') = M(v')$,
- $\forall p \in P, M'(v)(p) = M(v)(p) - W^-(p, t) + W^+(p, t) +$ $\Sigma_{e \in E''} W^+(p, A(e), K(t, A(e)))$.

Definition 13. *The* firing *of an abstract transition t from a node v of an extended marking $tr = \langle V, M, E, A\rangle$ leads to the extended marking $tr' = \langle V', M', E', A'\rangle$ (denoted by $tr\xrightarrow{t,v}tr'$) if and only if:*

Let v' be a fresh identifier,

- $\forall p \in P, M(v)(p) \geq W^-(p, t)$,
- $\forall q \in Q$ *s.t.* $W^*(q, t)$ *is defined,* $M(v)(q) = W^*(q, t)$,
- $V' = V \cup \{v'\}$, $E' = E \cup \{(v, v')\}$,
- $\forall e \in E, A'(e) = A(e)$, $A'((v, v')) = t$,
- $\forall v'' \in V \setminus \{v\}, M'(v'') = M(v'')$,
- $\forall p \in P, M'(v)(p) = M(v)(p) - W^-(p, t)$,
- $M'(v') = \Omega(t, M(v)_{|Q})$.

Definition 14. *The* firing *of a cut step τ_i from a node v of an extended marking $tr = \langle V, M, E, A\rangle$ leads to the extended marking $tr' = \langle V', M', E', A'\rangle$ (denoted by $tr\xrightarrow{\tau_i,v}tr'$) if and only if:*

$M(v) \in \Upsilon_i$ and if v is the root of the tree then $tr' = \perp$, otherwise:

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

Let $tr \xrightarrow{s,v} tr'$ be a firing, then s is said to be an *enabled* step in v, denoted by $tr \xrightarrow{s,v}$. Other notations about firing sequences and reachability in SRPNs are carried over RPNs.

Remarks:

- $i \in I$ represents a possible effect (see the domain of function W^+) consecutive to a subtree pruning. There are two ways to prune a subtree: by a cut step when the ordinary marking of the subtree root belongs to Υ_i or by the firing of an elementary transition t which interrupts t', the transition labelling the edge to the subtree (i.e. $K(t, t') = i$). When Υ_i is the empty set this means that the index i can only be the effect of an interrupt.
- The bounding constraint on places of Q is implicitly taken into account by requiring that tr' introduced in definitions 12, 13 and 14 is an extended marking (see the second item of definition 11).
- Since in the previous definitions, an elementary step requires the tree is not empty, the extended marking $\perp$ is dead.
- Inhibitor arcs connected to bounded places could be easily simulated by test arcs. Due to the boundedness requirement, this does not lead to undecidability for the reachability problem.

We now justify the introduction of test arcs in the model. In ordinary Petri nets, a test arc on a bounded place can be simulated by the introduction of a *complementary* place which records the difference between the bound and the current marking such that an input arc from the original place generates an output arc to this place and *vice versa*. Then, a test arc is transformed in two loops around the two places, respectively labelled by the tested value and its difference with the bound.

Such a construction is no more valid when the test arc is connected to an abstract transition since then the consuming and producing of tokens are performed in two distinct steps. Indeed, the effect of W^+ is delayed until a cut step occurs in the new thread or an elementary transition is fired which interrupts the thread subtree initiated by the abstract transition.

Furthermore, note that according to semantics $\forall p \in P, \forall t \in T$ when defined $W^*(p,t) \geq W^-(p,t)$ since otherwise the transition t will never be enabled.

Although the semantics of SRPN and RPN are different, any marked SRPN (N, tr_0) can be simulated by a marked RPN (N', tr'_0) in such a way that the standard equivalent relations are fulfilled by the two nets (bisimulation, language equivalence, reachability graph isomorphism, etc). The construction is straightforward, so we informally describe it. First N' is a copy of N. Then a control place is added to N' and is both an input and a output place of every transition. The starting marking of any abstract transition is equal to the original one plus a token in this control place. Similarly, tr'_0 is a copy of tr_0 with the control place marked only in the active node.

3.2 An illustrative example

The net of figure 7 illustrates the characteristic features of RPNs. The items which are absent in SRPNs are represented graphically as follow:

- a place of Q by a double border circle whose bound is indicated in brackets after its name and
- a test arc between a place p and a transition t by a simple line. This line is labelled with $W^*(p,t)$ when $W^*(p,t) \neq 1$.

The name of an elementary transition t is followed by the set $\{t'\langle i\rangle \mid K(t,t') = i\}$ when this set is non empty. The starting marking of an abstract transition is defined by a mapping from a finite domain to the set of ordinary markings. So the modeler could define it by an enumeration. However for our examples, we adopt a more concise way to specify it. Given an abstract transition t, its starting marking mapping is indicated in a frame with the following syntax: $\Sigma_{p \in P}[\Omega_{p,t}] \cdot p$ where $\Omega_{p,t}$ is an arithmetical expression involving the variables $\{q \mid q \in Q\}$. Given a current submarking m on Q and a place p the value of $\Omega(t,m)(p)$ is obtained by evaluating the expression where any variable q is replaced by $m(q)$. Here again, when $\Omega_{p,t}$ is null the term $[\Omega_{p,t}] \cdot p$ is omitted and is abbreviated to p when $\Omega_{p,t} = 1$. For instance, $[p_{rec} - 1] \cdot p_{rec} + p_{int}$ means that the starting marking m of a node created by firing t_{fork} will be defined by $\forall p \in P \setminus \{p_{int}, p_{rec}\}, m(p) = 0$, $m(p_{int}) = 1$ and $m(p_{rec}) = v - 1$ where v is the number of tokens in place p_{rec} in the marking of the node where the firing occurs.

The net in figure 7 shows the modelling of similar transactions performed by a remote server. A transaction is started by the firing of the transition t_{start}. The status of the server is described by the places *On* and *Off*. Thus, the firing of t_{start} is controlled by a test arc connected to the place *On*. A transaction may either commit

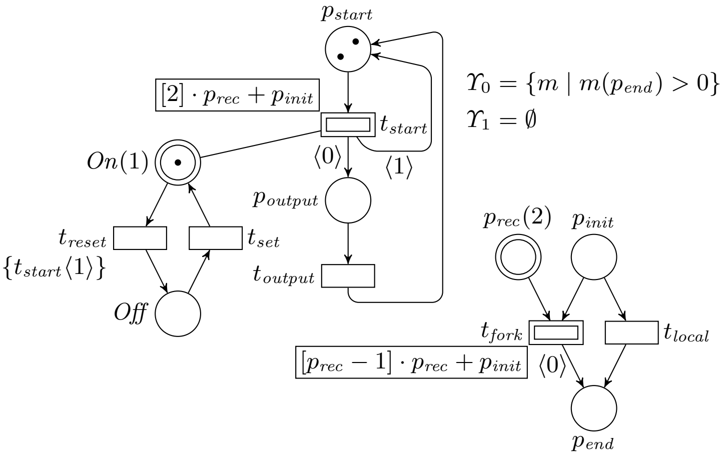


Fig. 7 a simple recursive Petri net

(represented by the index $\langle 0 \rangle$) or abort when the server is reset (represented by the index $\langle 1 \rangle$). Since there are two ways to terminate a transaction, I includes two items ($I = \{0, 1\}$). In the first case, the termination of a transaction is indicated by a token in p_{end} (see Υ_0). Then the cut step puts a token in the place p_{output} which controls the transition t_{output} modelling the transmission of the result. In the second case, the abortion is modelled by the transition t_{reset} which interrupts t_{start} with the index $\langle 1 \rangle$. This firing produces a token in place p_{start} meaning that the transaction does not provide a result. Υ_1 is empty since the abortion is never triggered by the execution of the transaction but only when the server is reset. If the modelling of faults would be required, it could be introduced by additional places and modifying Υ_1.

When started, a transaction may proceed locally by firing t_{local} or starts a new process by firing t_{fork}. The place p_{rec} controls the maximal number of nested forks. The starting marking of t_{start} expresses that there will be at most two forks per transaction. Note that a thread initiated by the transition t_{fork} has an initial marking with one token less in the place p_{rec} than its initiator. The abortion process realised by the transition t_{reset} applies on the whole transactions i.e. in the root of the reached extended markings. It is easy to see that in another node the transition t_{reset} is never enabled.

Notice that the bounded places are On and p_{rec}. Thus, this net fulfills the conditions of def. 10: the single test arc is connected to On and starting marking of t_{fork} depends exclusively on p_{rec}.

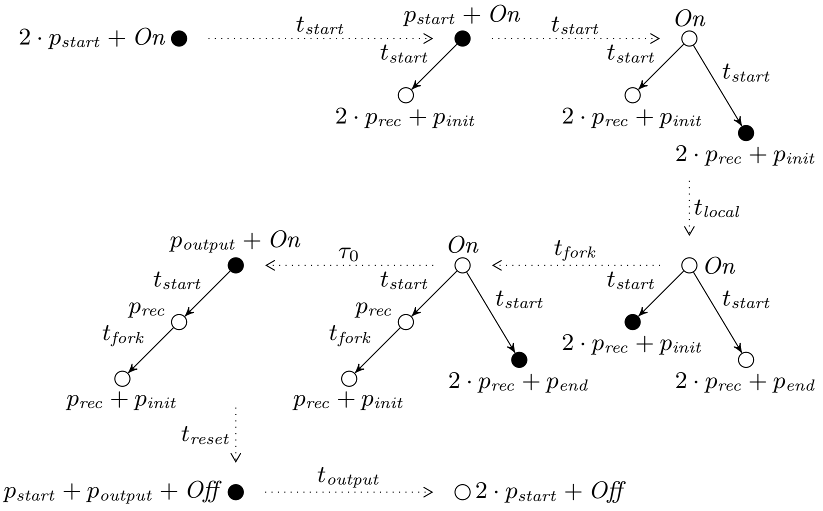


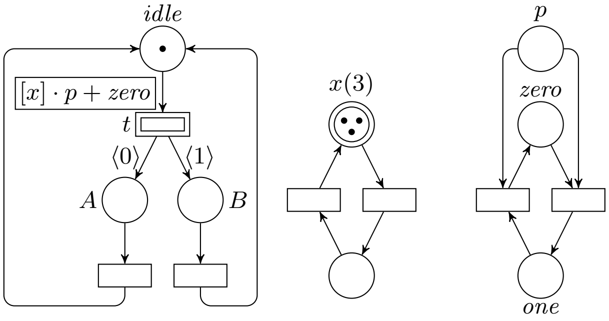
Fig. 8 a firing sequence

The initial state of the net is a tree reduced to a single node with two tokens in p_{start} corresponding to two transactions to begin and one token in On indicating that the server is operational. A firing sequence of this marked RPN is presented in the figure 8. The arcs of the trees composing the visited extended markings are labelled by the abstract transition t_{start} for the outgoing arcs from the root and by t_{fork} for the other ones. The black node of an extended marking denotes the thread initiator of the current step. Let us notice that each firing of an abstract transition leads to the creation of a new node in the tree whereas the firing of the cut step prunes the subtree of the root represented in the figure as its right branch. Moreover the firing of the elementary transition t_{reset} prunes its remaining subtree by the interrupt mechanism.

3.3 Expressivity of RPNs

Modelling of asynchronous remote procedure calls: The procedure we model has a non negative integer as input and returns whether this parameter is odd or even. During the computation the caller is not suspended and for instance may modify the variable which has been transmitted (by value) to the procedure.

The caller consists in two processes: the first one, modelled by the subnet on the left part of the figure 9, iteratively calls the procedure whereas the second one, modelled by the subnet on the central part,



$$\Upsilon_0 = \{m \mid m(p) = 0 \wedge m(zero) = 1\}$$
$$\Upsilon_1 = \{m \mid m(p) = 0 \wedge m(one) = 1\}$$

Fig. 9 an asynchronous remote procedure call example

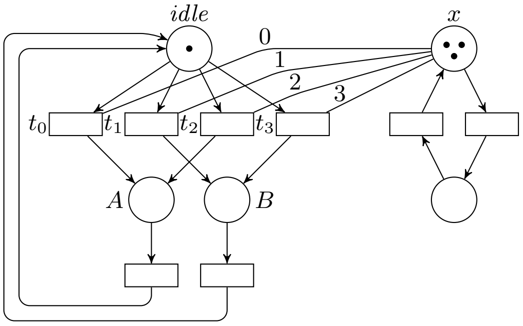
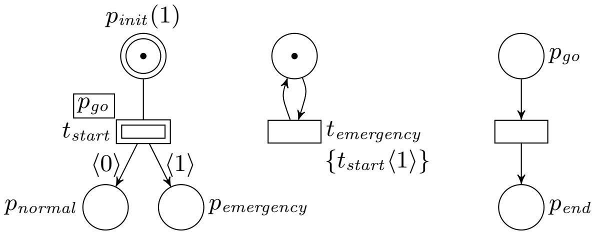


Fig. 10 a Petri net modelling of the parity test

increments or decrements the value of variable x in the interval $[0, 3]$. The current value of the variable x is denoted by the marking of the eponymous place. The procedure call is represented by the firing of the abstract transition t. Its starting marking “copies” the value of x in place p. The result returned by the procedure is specified by the index of the reached final marking set labelling the output arcs of t: the post condition of t produces a token either in the place A or in B with respect to the final marking reached by the thread initiated by t.

The procedure is modelled by the subnet on the right part of figure 9. This subnet determines whether the starting marking of the place p is even (by reaching a marking of Υ_0) or not (by reaching a marking of Υ_1). Notice that the process which updates x is not suspended during the call and then the marking of x can evolve between the firing of t and the firing of the corresponding cut step.



$$\Upsilon_0 = \{m \mid m(p_{end}) = 1\}$$
$$\Upsilon_1 = \emptyset$$

Fig. 11 an emergency reaction

The modelling of this pattern by a Petri net raises the problem consisting in assigning the marking of a place to the marking of another place (representing the transmission of a parameter). To the best of our knowledge, even when places are bounded, there is no structural solution, i.e. the nets modelling such a pattern depend on the bounds of the places.

The figure 10 presents one possible modelling of the asynchronous remote procedure call by a Petri net. Since the place x is bounded, we have used the test arcs in order to obtain a concise model. With the help of a complementary place, each test arc connected to some t_i could be replaced by two loops connecting, on the one hand, t_i and x and, on the other hand, t_i and the complementary place of x.

Our solution uses $\mathcal{B}(x) + 1$ transitions (one per possible marking of x) to return the result of the procedure. It is an open question even in this particular case whether a Petri net may be designed independently of the bound of x (here 3).

Modelling of emergency situations: The activity scheduling of an embedded critical system depends on the behaviour of the environment. When an emergency situation is encountered, non critical activities are aborted in order to react to this situation. Let us describe how to model such a system with RPNs.

The net of figure 11 represents a system which can initiate any number of tasks by firing t_{start}. Each task is abstracted by the trivial subnet at the right of the figure. At any time, transition $t_{emergency}$ is enabled in the root of the extended marking and its firing interrupts all the subtrees initiated at this level. The tasks may also achieve their execution. The two kinds of termination are distinguished by the corresponding indexes of the RPN similarly to the net of figure 7. Assume a current extended marking with n executions of tasks, i.e. a tree where the root has n direct successors which are the leaves of the tree. Note that there are $n+1$ such extended markings (related to the number of tasks where p_{end} is marked). Assume that $t_{emergency}$ is fired. Then whatever the extended marking among them, the reached one is the same. Since n may be arbitrarily chosen, this means that the input degree of states of this RPN is unbounded. As already discussed, such a behaviour is impossible with standard Petri nets.

3.4 Analysis of RPNs

Decidability of the reachability and boundedness problems: First, we recall that, in a Petri net N, the problem whether a semilinear set of markings SL is reachable from an initial one m_0 (i.e. whether at least one marking of SL is reachable from m_0) is reducible to the standard reachability problem and thus decidable. The principle of the reduction is the following one:

- One successively tests the reachability of the linear sets composing the semilinear set.
- Let a linear set L be defined by a marking m and a finite set of markings $\{m_1, \ldots, m_k\}$ such that $L = \{m' \mid \exists(\lambda_1, \ldots, \lambda_k) \in \mathbb{N}^k,\ m' = m + \sum_{i=1,\ldots,k} \lambda_i \cdot m_i\}$. One builds a net N_L from N and L. It includes N as a subnet and k supplementary transitions in order to consume the tokens corresponding to $m_1, \ldots, m_k$.
- It is straightforward that L is reachable from m_0 in N iff m is reachable from m_0 in N_L.

We will assume that all the bounded places have the same bound b i.e. $\forall q \in Q, \mathcal{B}(q) = b$ since first it alleviates notations and second all subsequent proofs can be straightforwardly extended to the general case. We note $B = [0, b]$ and so $\mathcal{B}[Q] = B^Q$.

Notations:

- Let $t \in T$ and $P' \subseteq P$ then $W^-(P', t)$ is the vector belonging to $\mathbb{N}^{P'}$ defined by $\forall p \in P', W^-(P', t)(p) = W^-(p, t)$,
- Let $t \in T_{el}$ and $P' \subseteq P$ then $W^+(P', t)$ is the vector belonging to $\mathbb{N}^{P'}$ defined by $\forall p \in P', W^+(P', t)(p) = W^+(p, t)$,
- Let $t \in T_{ab}, i \in I$ and $P' \subseteq P$ then $W^+(P', t, i)$ is the vector belonging to $\mathbb{N}^{P'}$ defined by $\forall p \in P', W^+(P', t, i)(p) = W^+(p, t, i)$,
- Let $t \in T$ and $Q' \subseteq Q$ then $W^*(Q', t)$ is the partial function from Q' to $\mathbb{N}$ such that $\forall q \in Q', W^*(Q', t)(q)$ is defined iff $W^*(q, t)$ is defined and then $W^*(Q', t)(q) = W^*(q, t)$.

The procedure which decides whether an extended marking is reachable in a marked RPN involves two stages:

- The first stage is independent from the initial and the final extended markings. It consists in deciding whether an RPN starting with an extended marking reduced to a node with initial marking $\Omega(t, m)$ reaches a state where the marking of the root belongs to Υ_i. In other words, we check whether the firing of an abstract transition t with m as the current submarking on Q is able to produce a τ_i step. We call this problem *the closability problem*. We restrict m to be a submarking on Q compatible with t since otherwise t is never enabled with such an m.
- The second step works top down on the trees associated with the initial and the final states. Given a potential firing sequence from the initial to the final states, it predicts the corresponding behaviour of each subtree (i.e. the tree rooted in a son of the root) of the two roots: either a subtree of the initial extended marking has disappeared or it has been transformed into a subtree of the final state. The remaining subtrees of the final state must have been created during the firing sequence. For each potential behaviour, it checks whether a firing sequence exists between the two roots and applies recursively a similar procedure on the reachability problems induced by its hypothesis on the subtrees. Its answers positively iff at least one behaviour is valid (see fig. 12 for two examples of possible behaviours).

All the elementary steps of this procedure are based on the same pattern: the building of an ordinary Petri net and of a reachability problem equivalent to the elementary problem to be decided. We proceed to solve the closability problem. The next definition introduces some sets of tuples (t, m, i) meaning that the firing the abstract transition t when the current marking of the node over Q is m may be followed by a cut step τ_i fired in the node created by the firing of t. We will call a corresponding firing sequence a *closing sequence* of (t, m, i). *Closable* is the set of all such tuples. Its subset $\mathit{Closable}_0$ requires that along at least one closing sequence there will be no cut step whereas $\mathit{Closable}_n$, with $n > 0$, requires that along at least one closing sequence, there will be no cut step in a node whose depth in the tree is greater than n (with depth 1 for the root). Below, we define these sets with an equivalent inductive definition.

Definition 15. *Let $N = \langle P, \mathcal{B}, T, I, W^-, W^*, W^+, \Omega, \Upsilon, K\rangle$ be a recursive Petri net, then:*

- *Closable is the set of tuples (t, m, i) with $t \in T_{ab}, m \in B^Q$ compatible with t, $i \in I$ such that there exist $tr \neq \perp$ an extended marking*

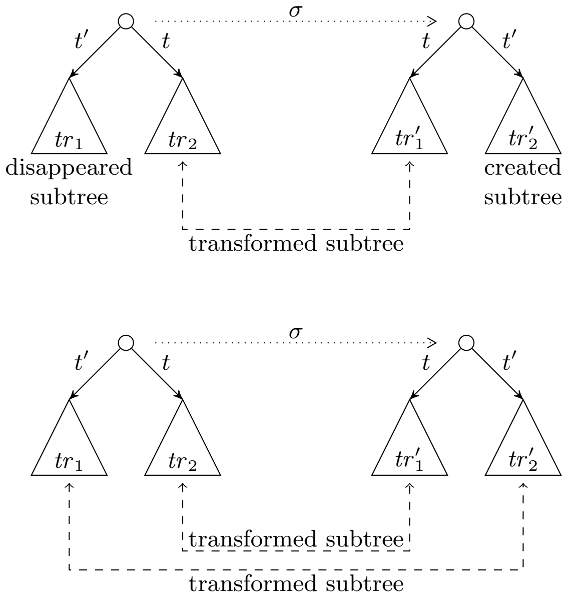


Fig. 12 two possible matchings of subtrees

and σ a firing sequence with:

$\lceil \Omega(t,m) \rceil \xrightarrow{\sigma}_N tr$ *and $v_0(tr)$, the root of tr, belongs to Υ_i*

such a sequence σ is said a closing sequence *w.r.t. (t,m,i).*

- *$Closable_0 \subseteq Closable$ is such that a tuple $(t,m,i) \in Closable_0$ iff there exists a closing sequence σ w.r.t. (t,m,i) which does not include any cut step occurring in a direct successor of the root.*
- *$Closable_{n+1} \subseteq Closable$ is such that a tuple $(t,m,i) \in Closable_{n+1}$ iff there exists a closing sequence σ w.r.t. (t,m,i) where any cut step τ_j, occurring in a direct successor of the root, corresponds to the firing of an abstract transition t' with a current submarking m' on Q and $(t',m',j) \in Closable_n$.*

We illustrate this definition on the net of the figure 13. The set of abstract transitions is $\{t, t'\}$; Q, the subset of bounded places, is the singleton $\{q\}$ where the bound of q is 3 and $I = \{0, 1\}$. Thus *Closable* $\subseteq \{t, t'\} \times \{(0), (1), (2), (3)\} \times \{0, 1\}$. Moreover, the tuples $(t', (k), i)$ such that $k > 0$ and the tuples $(t, (0), i)$ whatever i are excluded of *Closable* since their submarking is not compatible with their transition.

We indicate below both the sets $Closable_n$ and some corresponding closing sequences.

- $Closable_0 = \{(t', (0), 1)\}$

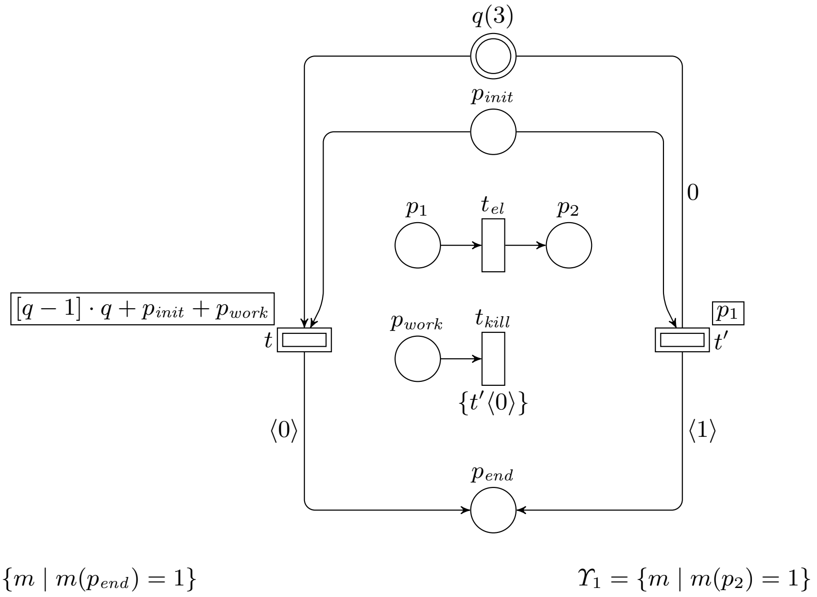
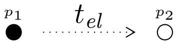


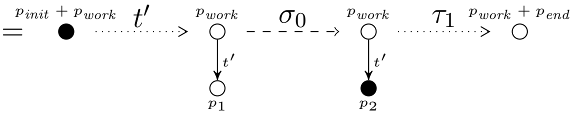
Fig. 13 a RPN N

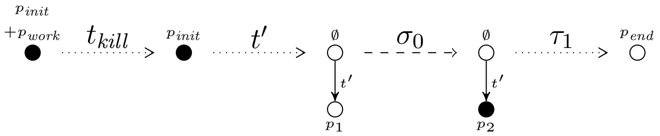
Let $\sigma_0 =$ 

Since $\Omega(t',(0)) = p_1$ and the marking $p_2 \in \Upsilon_1$, σ_0 is a closing sequence for $(t',(0),1)$.

Note that $(t,(k),i)$ (whatever k and i) does not belong to $Closable_0$ since, due to the definition of $\Omega(t,(k))$, p_2 can never be marked and p_{end} can only be marked by firing one of the abstract transitions t and t' later followed by a cut step.

- $Closable_1 \setminus Closable_0 = \{(t,(1),0)\}$

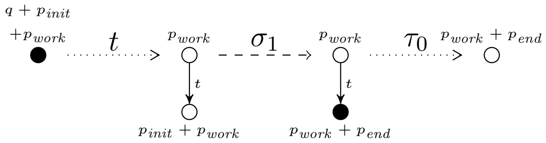
Let $\sigma_1 =$ 

and $\sigma'_1 =$ 

Then σ_1 and σ'_1 are closing sequences of $\{(t,(1),0)\}$. Note that both sequences include the sequence σ_0 in order to prune the subtree created by the firing of t'.

- $Closable_2 \setminus Closable_1 = \{(t,(2),0)\}$

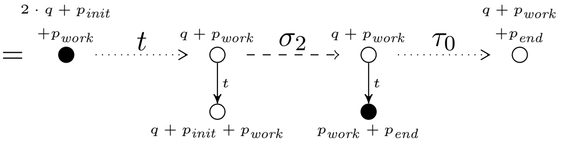
Let $\sigma_2 =$



Then σ_2 is a closing sequence of $\{(t, (2), 0)\}$.

- *Closable*$_3 \setminus$ *Closable*$_2 == \{(t, (3), 0)\}$

Let $\sigma_3 =$



Then σ_3 is a closing sequence of $\{(t, (3), 0)\}$.

- *Closable*$_4 \setminus$ *Closable*$_3 = \emptyset$ and *Closable*$_3 =$ *Closable*

As pointed out by the previous example, the following lemma is a direct consequence of the definition 15 and is the basis for an iterative computation of *Closable* $= \bigcup_{n \geq 0}$ *Closable*$_n$.

Lemma 4. *Let $n_0 \in \mathbb{N}$ such that Closable$_{n_0+1}$ = Closable$_{n_0}$. Then, $\forall n > n_0$, Closable$_n$ = Closable$_{n_0}$. Consequently, since Closable is a finite set, $\exists n_0$ such that Closable = Closable$_{n_0}$ with $\forall 0 \leq n \leq n_0$, Closable$_n \setminus$ Closable$_{n-1} \neq \emptyset$ (using the convention that Closable$_{-1} = \emptyset$).*

The next lemma is the crux of the decidability of the reachability problem. Notice that, due to lemma 4, it already implies that *Closable* is computable.

Lemma 5. *Let $N = \langle P, \mathcal{B}, T, I, W^-, W^*, W^+, \Omega, \Upsilon, K \rangle$ be a recursive Petri net and $n \in \mathbb{N} \cup \{-1\}$. Assume that Closable$_n$ is known then Closable$_{n+1}$ is computable.*

Proof. This proof is divided in two parts. Given a recursive Petri net N, n such that*Closable*$_n$ is known and a tuple $(t_a, m_a, i) \notin$ *Closable*$_n$, first we build a marked Petri net (N^*, m_0^*) (depending on N, $\Omega(t_a, m_a)$ and *Closable*$_n$) and a semilinear set Υ_i^* of its markings (depending on Υ_i). Then, we prove that $(t_a, m_a, i) \in$ *Closable*$_{n+1}$ iff Υ_i^* is reachable from m_0^* in N^*.

Building of (N^*, m_0^*) and Υ_i^*: (N^*, m_0^*) is obtained as the synchronised product of a marked Petri net (N', m'_0) and of a finite automaton A. The net N' mimics the behaviour of N at the root level when the constraints relative to the places of Q are not taken into account and when all tuples are assumed to belong to *Closable*. The finite automaton A mimics the behaviour of N at the root level when the constraints relative to the places of $\overline{Q}$ are not taken into account. Furthermore in A, the firing of an abstract transition t, whenever the current marking over Q at the root level is m and leading to a cut step τ_j, requires that $(t, m, j) \in \mathit{Closable}_n$.

Let us briefly recall how the synchronized product of a marked Petri net and a finite automaton is performed:

- Its places are the ones of the net and the states of the automaton.
- The initial marking of places of the net is unchanged, the place corresponding to the initial state of the automaton contains a unique token and the places corresponding to the other states are unmarked.
- There is a transition $(t, s \xrightarrow{t} s')$ for every pair composed by t a transition of the net and $s \xrightarrow{t} s'$ a transition of the automaton whose label is the name of the net transition. Its inputs are the inputs of the net transition completed by s (with valuation one) and its outputs are the outputs of the net transition completed by s' (with valuation one).

The main property of the synchronized product that we will use is the following one:
there is a firing sequence $(t_0, s_0 \xrightarrow{t_0} s_1) \ldots (t_n, s_n \xrightarrow{t_n} s_{n+1})$ in the synchronized product iff there is a firing sequence $t_0 \ldots t_n$ in the Petri net and a path $s_0 \xrightarrow{t_0} s_1 \ldots \xrightarrow{t_{n-1}} s_n \xrightarrow{t_n} s_{n+1}$ in the automaton.

Construction of (N', m'_0)*:*
In the following, $T_{ia} = \bigcup_{t \in T_{el}} K(t)$ denotes the set of interruptible abstract transitions.

Notice that $N' = \langle P', T', W'^-, W'^+ \rangle$ does not depend on $\mathit{Closable}_n$.

The set of places of N' is composed by $\overline{Q}$ (whose incidence will be computed from the one of their corresponding places in N) and a set of additional control places:

$$P' = \overline{Q} \cup \{p_{t,t'} \mid t \in T_{ab}, t' \in K(t)\} \cup \{p_t, \overline{p_t} \mid t \in T_{el}\} \cup$$
$$\{p_{t,i} \mid t \in T_{ab} \setminus T_{ia}, i \in I\}$$

The arcs connected with these control places will be presented with the transitions of N'. Notice that the places of Q are not represented in N' but their effect is taken into account in the automaton A.

The transitions of N' are the following ones:

$$T' = \{t^- | t \in T_{ab}\} \cup (T_{ia} \times I) \cup \{(t,i)^-, (t,i)^+ | t \in T_{ab} \setminus T_{ia}, i \in I\} \cup$$

$$\{(t,t')^-, (t,t')^+ | t \in T_{ia}, t' \in K(t)\} \cup \{t^-, t^+ | t \in T_{el}\}$$

We simultaneously describe the meaning of each group of transitions and the incidence matrices:

- for each $t \in T_{ab}$, a transition t^- which simulates the firing of the abstract transition not followed either by a cut step or an interrupt which closes the subtree initiated by t. t^- consumes the tokens of $W^-(\overline{Q}, t)$:
  $\forall t \in T_{ab}, \forall p \in \overline{Q}, W'^-(p, t^-) = W^-(p,t), W'^+(p, t^-) = 0$
- for each couple $t \in T_{ia}, i \in I$, a transition (t,i) which simulates the firing of t, immediately followed by the cut step τ_i:
  $\forall (t,i) \in T_{ia} \times I, \forall p \in \overline{Q}$,
  $W'^-(p,(t,i)) = W^-(p,t), W'^+(p,(t,i)) = W^+(p,t,i)$
  As will be shown later, the closing sequence, when it exists, can be chosen such that every cut step related to the firing of an interruptible abstract transition occurs immediately after this firing.
- for each couple $t \in T_{ab} \setminus T_{ia}, i \in I$, two transitions $(t,i)^-$ and $(t,i)^+$. The transition $(t,i)^-$ simulates the firing of t by consuming the tokens required to fire t and producing a token in a control place $p_{t,i}$. The transition $(t,i)^+$ simulates the subsequent cut step τ_i consuming a token in $p_{t,i}$ and producing the tokens associated with i^{th} termination mode of t:
  $\forall t \in T_{ab} \setminus T_{ia}, \forall i \in I, \forall p \in \overline{Q}$,
  $W'^-(p,(t,i)^-) = W^-(p,t), W'^+(p,(t,i)^+) = W^+(p,t,i)$,
  $W'^+(p_{t,i},(t,i)^-) = 1, W'^-(p_{t,i},(t,i)^+) = 1$
  Along the simulating sequence of a closing one, a token in $p_{t,i}$ corresponds to a subtree created at the root level by a firing of the abstract transition t that will be later followed a cut step. So we require that such places are unmarked in the final marking (see the definition of Υ_i^*).
- for a couple of transitions $t \in T_{ia}$, $t' \in K(t)$, two transitions $(t,t')^-, (t,t')^+$. The transition $(t,t')^-$ corresponds to a firing of t that will be interrupted by t'. Thus it consumes the input tokens of t and puts a token in a place $p_{t,t'}$ in order to control the interrupt process. The transition $(t,t')^+$ consumes this token and produces the tokens associated with $W^+(\overline{Q}, t, K(t',t))$; it is a part of the simulation of the firing of t':
  $\forall t \in T_{ia}, t' \in K(t), \forall p \in \overline{Q}$,
  $W'^-(p,(t,t')^-) = W^-(p,t), W'^+(p_{t,t'},(t,t')^-) = 1$,
  $W'^-(p_{t,t'},(t,t')^+) = 1, W'^-(p_{t'},(t,t')^+) = 1$,
  $W'^+(p,(t,t')^+) = W^+(p,t,K(t,t')), W'^+(p_{t'},(t,t')^+) = 1$
- for each transition $t \in T_{el}$, two transitions t^-, t^+. The firing of t , occurring in a closing sequence, is simulated by a firing sequence

beginning by t^- followed by firings of transitions $(t',t)^+$ (where such a firing corresponds to an interrupt which closes a subtree created by the firing of $t' \in K(t)$ at the root level) and finished by t^+. The transition t^- consumes the tokens consumed by t, a token of a control place $\overline{p_t}$ and puts a token in another control place p_t. Furthermore, $\overline{p_t}$ loops around every transition (i.e. it is an input and output of these transitions with corresponding valuations equal to 1) except the transitions $(t',t)^+$. So t^- freezes the remaining transitions; the control place p_t loops around the transitions $(t',t)^+$ enabling their firing if tokens are present in $p_{t,t'}$. The firing of t^+ consumes the token in p_t and produces a new token in $\overline{p_t}$ unfreezing the net:
$\forall t \in T_{el}, \forall p \in \overline{Q}$,
$W'^-(p,t^-) = W^-(p,t), W'^+(p,t^+) = W^+(p,t)$,
$W'^-(\overline{p_t},t^-) = 1, W'^+(p_t,t^-) = 1$,
$W'^-(p_t,t^+) = 1, W'^+(\overline{p_t},t^+) = 1$
$\forall t \in T_{el}, \forall t' \in T' \setminus (\{t^-,t^+\} \cup \{(t'',t)^+ \mid t'' \in K(t)\})$,
$W'^+(\overline{p_t},t') = 1, W'^-(\overline{p_t},t') = 1$

- For all other pairs $(p',t') \in P' \times T'$,
  $W'^-(p',t') = 0$ and $W'^+(p',t') = 0$.

The initial marking m'_0 of the net is the projection of $\Omega(t_a,m_a)$ on $\overline{Q}$ and a token in each place $\overline{p_t}$. All other places are unmarked:
$\forall p \in \overline{Q}, m'_0(p) = \Omega(t_a,m_a)(p)$
$\forall t \in T_{el}, m'_0(\overline{p_t}) = 1$ and $m'_0(p_t) = 0$
$\forall t \in T_{ab}, \forall t' \in T_{el}, m'_0(p_{t,t'}) = 0$
$\forall t \in T_{ab} \setminus T_{ia}, \forall i \in I, m'_0(p_{t,i}) = 0$

The construction of (N',m'_0) for the recursive net of figure 13 is given in the figure 14. m'_0 corresponds to the closability problem of (t,m,i) (for any m and any i): $m'_0(p_{init}) = m'_0(p_{work}) = 1$ due to the definition of $\Omega(t,m)$; $m'_0(\overline{p}_{t_{el}}) = m'_0(\overline{p}_{t_{kill}}) = 1$ and the other places are unmarked according to the construction. We do not describe all the transitions but, for instance, a firing of t' in N is simulated either by $(t',1)$ if the firing of t' is later followed by a corresponding cut step, either by t'^- if the subtree created by t' is still present in the final marking, or by a pair $(t',t_{kill})^-$, $(t',t_{kill})^+$ if the subtree created by t' will be later deleted by the firing of t_{kill} (which interrupts t').

We now give three firing sequences of (N',m'_0) in order to show how the simulation works:

- $t^-_{kill} \cdot t^+_{kill} \cdot (t',1)$ which corresponds to the simulation of σ'_1. We will show that this sequence will be accepted by the automaton A relative to $Closable_0$ and the closability of $(t,(1),0)$.

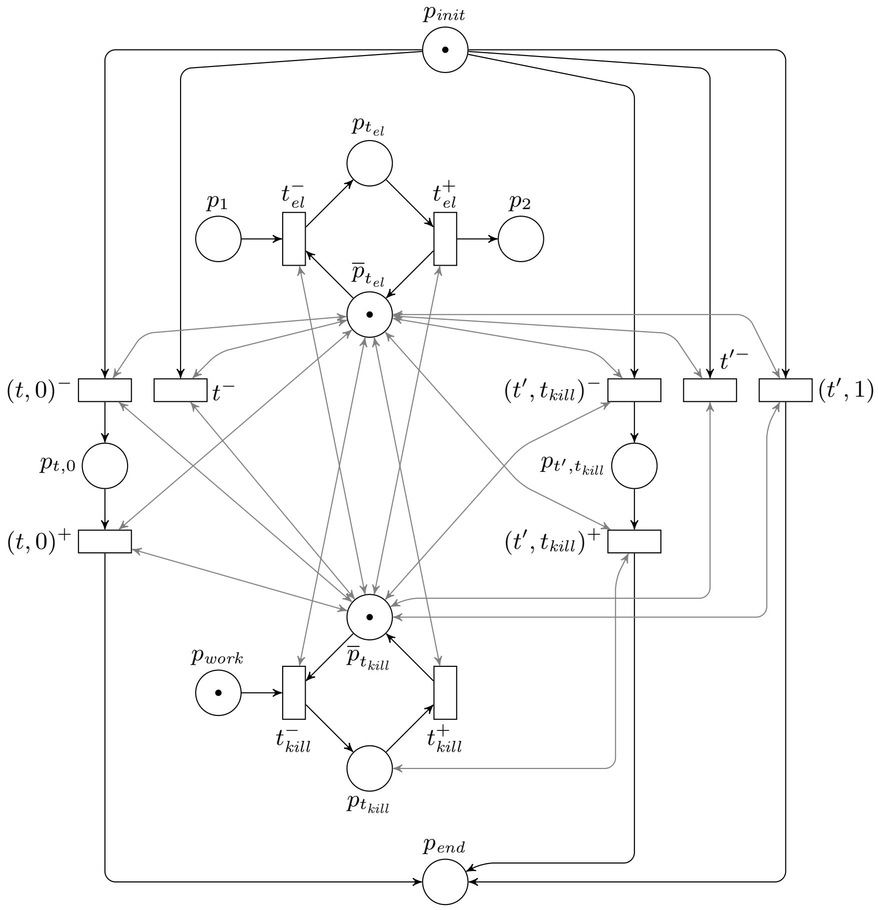


Fig. 14 the ordinary net (N', m'_0) corresponding to the net N of figure 13 and the closability problem for (t, m, i) (for any m and any i)

- $(t,0)^- \cdot (t,0)^+$ which corresponds to the simulation of both σ_2 and σ_3. We will show that the automaton A relative to *Closable*$_1$ and the closability of $(t,(2),0)$ accepts this sequence while the automaton A relative to *Closable*$_1$ and the closability of $(t,(3),0)$ discards it.
- $t'^- \cdot t_{kill}^- \cdot t_{kill}^+$ which cannot simulate any firing sequence of N since in N a firing of t' followed by a firing of t_{kill} must delete the subtree created by t' and then produce a token in p_{end}. Such a sequence will be discarded by the automaton A relative to any *Closable*$_i$ and the closability of any tuple.

Construction of A: The automaton $A = \langle S, \{\xrightarrow{t}\}_{t \in T'}, s_0 \rangle$, which depends on N, $\mathit{Closable}_n$ and the considered pair (t_a, m_a), is defined as follows.

Each state is a tuple $(m, (M_{tb})_{tb \in T_{ia}}, Tc)$ where:

- $m \in B^Q$ corresponds to the submarking on Q at the root level of the simulated extended marking of N,
- $M_{tb} \subseteq B^Q$ denotes the sets of submarkings on Q at the root level of the simulated extended marking of N encountered since the last firing of a transition $t' \in K(tb)$ (or since the initial marking if none has been fired),
- $Tc \subseteq T_{el}$ is the set of elementary transitions still allowed to fire at the root level of the simulated extended marking of N (i.e. not belonging to some $K(t)$ with $t \in T_{ab}$ when the firing of a transition t^- has previously occurred): $S = B^Q \times (2^{(B^Q)})^{T_{ia}} \times 2^{T_{el}}$.

The first component of the initial state is $\Omega(t_a, m_a)_{|Q}$, the second one is the vector of singletons associated with this submarking and the third one is T_{el}: $s_0 = (\Omega(t_a, m_a)_{|Q}, (\{\Omega(t_a, m_a)_{|Q}\})_{tb \in T_{ia}}, T_{el})$.

The transitions of the automaton are labelled by the transitions of N' and defined according to their interpretation. *These transitions are only defined when they lead to a new marking m' belonging to B^Q* and this condition will be implicit in the sequel. Given a state $(m, (M_{tb})_{tb \in T_{ia}}, Tc)$, we define below its output transitions:

- for each $t \in T_{ab}$, the transition t^- is enabled depending on m and $W^*(Q,t)$ and then leads to the state $(m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc \setminus K(t))$, where m' is defined from m by consuming the tokens specified by $W^-(Q,t)$:
  $\forall t \in T_{ab}$ s.t. m is compatible with t,
  let m' defined by $\forall p \in Q, m'(p) = m(p) - W^-(p,t)$ then
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{ta^-} (m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc \setminus K(t))$$
- for each $t \in T_{ia}$, $i \in I$, the transition (t,i) is enabled iff $(t,m,i) \in \mathit{Closable}_n$ and then loops around the state (since the marking of places in Q is not modified by the interruptible abstract transitions):
  $\forall t \in T_{ia}, \forall i \in I$ s.t. $(t,m,i) \in \mathit{Closable}_n$,
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{(t,i)} (m, (M_{tb})_{tb \in T_{ia}}, Tc)$$
- for each $t \in T_{ab} \setminus T_{ia}$, $i \in I$, the transition $(t,i)^-$ is enabled iff $(t,m,i) \in \mathit{Closable}_n$ and then leads to the state $(m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc)$, where m' is defined from m by consuming the tokens specified by $W^-(Q,t)$:
  $\forall t \in T_{ab} \setminus T_{ia}, \forall i \in I$ s.t. $(t,m,i) \in \mathit{Closable}_n$,

let m' defined by $\forall p \in Q, m'(p) = m(p) - W^-(p,t)$ then

$$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{(t,i)^-} (m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc)$$

- for each $t \in T_{ab} \setminus T_{ia}$, $i \in I$, $m \in B^Q$, the transition $(t,i)^+$ is enabled and leads to the state $(m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc)$, where m' is defined by producing the tokens specified by $W^+(Q,t,i)$:
  $\forall t \in T_{ab} \setminus T_{ia}, \forall i \in I$,
  let m' defined by $\forall p \in Q, m'(p) = m(p) + W^+(p,t,i)$,
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{(t,i)^+} (m', (M_{tb} \cup \{m'\})_{tb \in T_{ia}}, Tc)$$
- for each $t \in T_{ia}$ and $t' \in Tc \cap K(t)$, the transition $(t,t')^-$ is enabled depending on m and $W^*(Q,t)$ and then loops around the state (since the marking of places in Q is not modified by the interruptible abstract transitions):
  $\forall t \in T_{ia}$ s.t. m is compatible with t, $\forall t' \in Tc \cap K(t)$,
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{(t,t')^-} (m, (M_{tb})_{tb \in T_{ia}}, Tc)$$
- for each $t \in T_{ia}$, $t' \in Tc \cap K(t)$, the transition $(t,t')^+$ is enabled depending on the existence of $m' \in M_t$ compatible with t and then loops around the state:
  $\forall t \in T_{ia}, \forall t' \in Tc \cap K(t)$, if $\exists m' \in M_t$ compatible with t then
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{(t,t')^+} (m, (M_{tb})_{tb \in T_{ia}}, Tc)$$
- for each $t \in Tc$, the transition t^- is enabled depending on m, $W^*(Q,t)$ and $W^-(Q,t)$ and leads to $(m', (M_{tb})_{tb \in T_{ia}}, Tc)$ where m' is defined by consuming the tokens specified by $W^-(Q,t)$. Since m' is an intermediate submarking which does not necessarily correspond to a marking of the RPN, it is not added to the subsets $(M_{tb})_{tb \in T_{ia}}$:
  $\forall t \in Tc$ s.t. m is compatible with t,
  let m' defined by $\forall p \in Q, m'(p) = m(p) - W^-(p,t)$ then
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{t^-} (m', (M_{tb})_{tb \in T_{ia}}, Tc)$$
- for each $t \in Tc$, the transition t^+ is enabled and, denoting m' the marking after producing the tokens of $W^+(Q,t)$, leads to $(m', (M'_{tb})_{tb \in T_{ia}}, Tc)$ a new state where m' is added to any M_{tb} with a preliminary reset if tb belongs to $K(t)$:
  $\forall t \in Tc$, let m' defined by $\forall p \in Q, m'(p) = m(p) + W^+(p,t)$,
  $$(m, (M_{tb})_{tb \in T_{ia}}, Tc) \xrightarrow{t^+} (m', ((\{m'\})_{tb \in K(t)}, (M_{tb} \cup \{m'\})_{tb \in T_{ia} \setminus K(t)}), Tc)$$

All the possible automata A, corresponding to the net N of figure 13, have been factorized in figure 15. The thickness of an arrow denotes to which automata it belongs. The initial state of the automaton depends on the tuple (t, m, i) and more precisely, on $\Omega(t, m)$; we

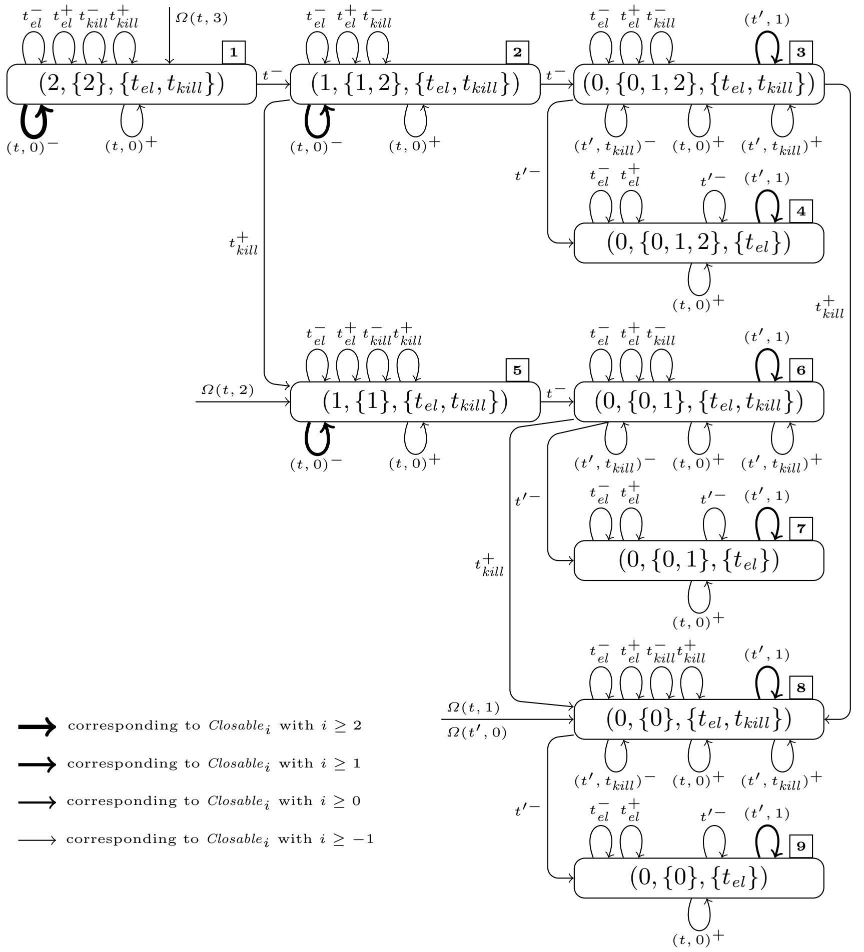

t_{el}^- t_{el}^+ t_{kill}^- t_{kill}^+
$\Omega(t,3)$
1
$(2,\{2\},\{t_{el},t_{kill}\})$
$(t,0)^-$ $(t,0)^+$
t^-
t_{el}^- t_{el}^+ t_{kill}^-
2
$(1,\{1,2\},\{t_{el},t_{kill}\})$
$(t,0)^-$ $(t,0)^+$
t^-
t_{el}^- t_{el}^+ t_{kill}^-
$(t',1)$
3
$(0,\{0,1,2\},\{t_{el},t_{kill}\})$
$(t',t_{kill})^-$ $(t,0)^+$ $(t',t_{kill})^+$
t'^-
t_{el}^- t_{el}^+ t'^- $(t',1)$
4
$(0,\{0,1,2\},\{t_{el}\})$
$(t,0)^+$
t_{kill}^+
t_{kill}^+
t_{el}^- t_{el}^+ t_{kill}^- t_{kill}^+
5
$\Omega(t,2)$
$(1,\{1\},\{t_{el},t_{kill}\})$
$(t,0)^-$ $(t,0)^+$
t^-
t_{el}^- t_{el}^+ t_{kill}^-
$(t',1)$
6
$(0,\{0,1\},\{t_{el},t_{kill}\})$
$(t',t_{kill})^-$ $(t,0)^+$ $(t',t_{kill})^+$
t'^-
t_{el}^- t_{el}^+ t'^- $(t',1)$
7
$(0,\{0,1\},\{t_{el}\})$
$(t,0)^+$
t_{kill}^+
t_{el}^- t_{el}^+ t_{kill}^- t_{kill}^+
$(t',1)$
8
$\Omega(t,1)$
$\Omega(t',0)$
$(0,\{0\},\{t_{el},t_{kill}\})$
$(t',t_{kill})^-$ $(t,0)^+$ $(t',t_{kill})^+$
t'^-
t_{el}^- t_{el}^+ t'^- $(t',1)$
9
$(0,\{0\},\{t_{el}\})$
$(t,0)^+$
corresponding to $Closable_i$ with $i \geq 2$
corresponding to $Closable_i$ with $i \geq 1$
corresponding to $Closable_i$ with $i \geq 0$
corresponding to $Closable_i$ with $i \geq -1$

Fig. 15 the possible automata A corresponding to the net N of figure 13

indicate it by a label on the initial arrows. In order to alleviate the notations in the figure, a submarking m over place q has been denoted by $m(q)$ instead of $(m(q))$. Notice that only reachable states have been represented on the figure.

As an illustration of the role of the automaton, we check whether the three sequences previously exhibited for the Petri net of the figure 14 are recognised by the automaton or not:

- $t_{kill}^- \cdot t_{kill}^+ \cdot (t',1)$
  This sequence (which corresponds to the simulation of σ_1') is recognised by the following path of the automaton relative to $Closable_0$ and the closability of $(t,(1),0)$:

$$\xrightarrow{\Omega(t,(1))}\boxed{8}\xrightarrow{t^-_{kill}}\boxed{8}\xrightarrow{t^+_{kill}}\boxed{8}\xrightarrow{(t',1)}\boxed{8}$$

- $(t,0)^- \cdot (t,0)^+$
  This sequence corresponds to the simulation of both σ_2 and σ_3. Starting from the initial state corresponding to $\Omega(t,(2))$, it is recognised by the following path of the automaton relative to *Closable*$_1$ and the closability of $(t,(2),0)$:
  $$\xrightarrow{\Omega(t,(2))}\boxed{5}\xrightarrow{(t,0)^-}\boxed{5}\xrightarrow{(t,0)^+}\boxed{5}$$
  But, starting from the initial state associated with $\Omega(t,(3))$, the automaton relative to *Closable*$_1$ and the closability of $(t,(3),0)$ discards it:
  $$\xrightarrow{\Omega(t,(3))}\boxed{1}\overset{(t,0)^-}{\dashrightarrow} \notin \mathit{Closable}_1$$
- $t'^- \cdot t^-_{kill} \cdot t^+_{kill}$
  This sequence cannot simulate any firing sequence of N and is discarded by the automaton relative to any *Closable*$_i$ and the closability of any tuple. For instance, the automaton relative to *Closable*$_0$ and the closability of $(t,(1),0)$ discards it:
  $$\xrightarrow{\Omega(t,(1))}\boxed{8}\xrightarrow{t'^-}\boxed{9}\overset{t^-_{kill}}{\dashrightarrow} \notin \mathit{Closable}_0$$

Specification of Υ_i^*: For all $i \in I$, we define Υ_i^* as the set of pairs of states of N' and A where the induced current marking on P belongs to Υ_i and the marking of the additional control places is identical to their initial marking. Formally:

$$\begin{aligned}\Upsilon_i^* = \{(m_1,(m_2,(M_{tb})_{tb\in T_{ia}},Tc)) \mid \\ m_1 \in \mathbb{N}^{P'} \wedge m_2 \in B^Q \wedge \forall tb \in T_{ia}, M_{tb} \subseteq B^Q \wedge Tc \subseteq T_{el} \wedge \\ (m_{1|\overline{Q}}, m_2) \in \Upsilon_i \wedge m_{1|P'\setminus\overline{Q}} = m'_{0|P'\setminus\overline{Q}}\}\end{aligned}$$

Let us illustrate this definition on the net of the figure 13. In case of

In N', Υ_0^* corresponds to markings m such that $m(p_{end}) = m(\overline{p}_{t_{kill}})$ $= m(\overline{p}_{t_{el}}) = 1$ and the other control places are unmarked.

In A, Υ_0^* corresponds to every state since place q is not involved in the specification of Υ_0.

Correctness of the previous construction
Let N be a RPN, t_a be an abstract transition, m_a be a submarking over Q compatible with t_a and i be a termination index. Let (N^*, m_0^*) be the marked Petri net and Υ_i^* be the semilinear set, both corresponding to the closability problem of (t_a, m_a, i) when *Closable*$_n$

is known. We now prove that there is a witness sequence relative to this problem in N iff there is sequence from m_0^* to Υ_i^* in N^*.
$(\Rightarrow)$

$$(t_a, m_a, i) \in \mathit{Closable}_{n+1} \Rightarrow \exists \sigma^*, \exists m_1 \in \Upsilon_i^* \text{ s.t. } m_0^* \xrightarrow{\sigma^*}_{N^*} m_1$$

Assume that there exist σ and tr with $M(v_0(tr)) \in \Upsilon_i$ such that $\lceil \Omega(t_a, m_a) \rceil \xrightarrow{\sigma}_N tr$ such that every firing of an abstract transition t in the root of the extended marking when the submarking over Q is m later followed by a corresponding cut step τ_j implies that $(t, m, j) \in \mathit{Closable}_n$. First, we show that we can transform the sequence in such a way that, *at the root level*, any firing of an interruptible abstract transition that will be closed by a cut step, occurs in σ immediately before this cut step. So assume that $\sigma = \sigma_1 \cdot t \cdot \sigma_2 \cdot \tau_j \cdot \sigma_3$ where the occurrence of t is a firing at the root level and τ_j is the corresponding cut step. Then we build a sequence σ' by firing immediately after t all the occurrences of firings and cut steps in the subtree created by t and then τ_j. The order of the remaining firings is unchanged. We claim that σ' is a firing sequence. Indeed in all subtrees of the root, the firing order is unchanged. Now at the root level, the cut step has been anticipated. This anticipation leads to intermediate markings where the submarking on Q is unchanged (due to the requirement about interruptible abstract transitions in definition 10) and the submarking on $\overline{Q}$ has increased (since a cut step can only produce tokens). Thus the new subsequence of occurrences of transitions at the root level is still a firing sequence.

After these transformations, the subsequence of σ' of firing occurrences at the root level is straightforwardly simulated by a sequence σ^* in N^* from m_0^*. To an abstract transition firing t which creates a subtree still present in the final state, corresponds a firing of t^-. To an interruptible abstract transition firing followed by a cut step corresponds the appropriate transition (t, j) (s.t. if m is the current marking on Q then $(t, m, j) \in \mathit{Closable}_n$ by definition of σ'). To a non interruptible abstract transition firing subsequently followed by a cut step corresponds the appropriate transition $(t, j)^-$ (s.t. if m is the current marking on Q then $(t, m, j) \in \mathit{Closable}_n$ by definition of σ') subsequently followed by the firing of the transition $(t, j)^+$. To an abstract transition firing t that will be interrupted by an elementary transition t' corresponds the firing of $(t, t')^-$. To an elementary transition t corresponds a sequence $t^- \cdot (t', t)^+ \ldots (t'', t)^+ \cdot t^+$ where the intermediate firings correspond to the abstract transitions firings interrupted by this occurrence.

It is also straightforward from the definition of Υ_i^* that the final marking belongs to this set.

$(\Leftarrow)$

$$(t_a, m_a, i) \in \mathit{Closable}_{n+1} \Leftarrow \exists\sigma^*, \exists m_1 \in \Upsilon_i^* \text{ s.t. } m_0^* \xrightarrow{\sigma^*}_{N^*} m_1$$

Let us assume that there exist σ^* and $m_1 \in \Upsilon_i^*$ s.t. $m_0^* \xrightarrow{\sigma^*}_{N^*} m_1$. Before building the corresponding sequence in N, we transform σ^*. We note that by construction of N^* and by definition of Υ_i^*, given $t \in T_{ia}$ and $t' \in K(t)$ there is the same number of occurrences of $(t,t')^-$ and $(t,t')^+$ and the k^{th} occurrence of the former precedes the k^{th} occurrence of the latter. However, between these occurrences there may be an occurrence of t''^+ with $t'' \in K(t)$. In such cases, due to the management of M_t in the automaton, we know that there is a submarking on Q which is compatible with t and encountered between the last occurrence of such t''^+ and the k^{th} occurrence of $(t,t')^+$. Thus we delay the k^{th} firing of $(t,t')^-$ until this submarking is reached. This can be done since the intermediate markings are increased on $\overline{Q}$ and unchanged on the other places of N^* except $p_{t,t'}$ which is decreased by one unit. However, this temporary decrease does not forbid any firing since this token will only be necessary to the k^{th} firing of $(t,t')^+$.

Now we build the corresponding sequence σ of N as follows. An occurrence of t^- with $t \in T_{ab}$ is substituted by t; the subtree created will still be present in the final marking due to the management of Tc in the automaton. An occurrence of (t,j) with m as the current marking on Q (thus $(t,m,j) \in \mathit{Closable}_n$) is substituted by the firing of t followed in the subtree by the closing sequence associated with (t,m,j) and finally by the cut step τ_j. An occurrence of $(t,j)^-$ with m as the current marking on Q (thus $(t,m,j) \in \mathit{Closable}_n$) is substituted by the firing of t followed in the subtree by the closing sequence associated with (t,m,j). We note that by construction of N^* and by definition of Υ_i^*, given $t \in T_{ab} \backslash T_{ia}$ and $j \in I$ there is the same number of occurrences of $(t,j)^-$ and $(t,j)^+$ and the k^{th} occurrence of the former precedes the k^{th} occurrence of the latter. The k^{th} occurrence of $(t,j)^+$ is substituted by the cut step τ_j in the subtree initiated by the firing of t corresponding to the k^{th} of $(t,j)^-$. An occurrence of $(t,t')^-$ is substituted by the firing of t. An occurrence of t^+ is substituted by the firing of t. The other occurrences are skipped.

With the initial transformation of the sequence σ', we obtain that the firing of $t \in T_{ia}$ in σ corresponding to the k^{th} occurrence of $(t,t')^-$ creates a subtree that will be destroyed by the firing of t' corresponding to the subsequence in σ', $t'^- \ldots (t,t')^+ \ldots t'^+$ where the k^{th} occurrence of $(t,t')^+$ appears. Thus at the root level the behaviour of σ is similar to σ^* and reaches a marking of Υ_i. □

Algorithm 2: Closable

```
$Comp = \emptyset$;
$Possible = \{(t,m) \in T_{ab} \times B^Q \mid m$ is compatible with $t\}$;
$Pot = Possible \times I$;
Build $N'$ and $\{\Upsilon_i^*\}_{i \in I}$;
repeat
    $New = \emptyset$;
    Build $A_{Comp}$ and then $N^*$;
    foreach $(t,m) \in Possible$ do
        Build $m_0^*$ depending on $(t,m)$;
        foreach $i$ s.t. $(t,m,i) \in Pot \setminus Comp$ do
            /* $ElemDecide$ returns true iff $\Upsilon_i^*$ is reachable from
               $m_0^*$ in the ordinary Petri net $N^*$                    */
            if $ElemDecide(N^*, m_0^*, \Upsilon_i^*)$ then
                $New = New \cup \{(t,m,i)\}$;
            end
        end
    end
    $Comp = Comp \cup New$;
until $(New == \emptyset)$ ;
return $Comp$;
```

Proposition 1. *Let $N = \langle P, \mathcal{B}, T, I, W^-, W^*, W^+, \Omega, \Upsilon, K \rangle$ be a recursive Petri net, then Closable is computable.*

Proof. Using lemmas 4 and 5, an iterative construction of *Closable* is immediate. The algorithm 2 summarises the decision procedure. □

Theorem 3. *The reachability problem is decidable for RPNs.*

Proof. Let N be an RPN and tr_0 and tr_1 two extended markings of N.

First, assume that $tr_0 = \perp$ then one checks whether $tr_1 = \perp$.

Now assume that $tr_0 \neq \perp$ and $tr_1 = \perp$. Then the last step of a hypothetical sequence from tr_0 to tr_1 is a cut step which means that, before this step, the marking at the root level belongs to some Υ_i. So we design the algorithm 3 which decides whether starting from tr_0, one can reach $\perp$ by a sequence ended by a cut step τ_i. This procedure looks for a sequence depending on the possible futures of every subtree rooted in $v_0(tr_0)$. Possible futures are represented by the set *Choices* and correspond to the three potential behaviours for every subtree tr_j^0 (with t_j^0 being the abstract transition labelling the edge to it). Let $ch \in Choices$ be such a choice:

- $ch(j) = k \in I$ means that tr_j^0 has disappeared in the penultimate state by the firing of a cut step τ_k.

- $ch(j) = t' \in K(t_j^0)$ means that tr_j^0 has disappeared in the penultimate state by the firing of an elementary transition denoted t'.
- $ch(j) = \top$ means that tr_j^0 is still present in the penultimate state.

We explain now a round of the outer loop corresponding to some $ch \in \mathit{Choices}$. First, it checks for every j such that $ch(j) \in I$ whether starting from tr_j^0, one can reach $\perp$ by a sequence ended by a cut step $\tau_{ch(j)}$. This is performed by a recursive call where the depth of the extended marking has been decreased by 1.

When all answers are positive, it builds an ordinary Petri net $N[ch]$, an initial marking $m_0[ch]$ and a semilinear set $\Upsilon_i[ch]$ such that:

$$\boxed{\begin{array}{c}\exists\sigma, \exists tr \text{ s.t. } tr_0 \xrightarrow{\sigma}_N tr,\ M(v_0(tr)) \in \Upsilon_i \text{ and the induced behaviour} \\ \text{of the subtrees rooted in } v_0(tr_0) \text{ is defined by } ch \\ \text{iff } \exists\sigma^*, \exists m_1 \in \Upsilon_i[ch] \text{ s.t. } m_0[ch] \xrightarrow{\sigma^*}_{N[ch]} m_1\end{array}}$$

Since $N[ch], m_0[ch]$ and $\Upsilon_i[ch]$ are similar to the net N^*, m_0^* and Υ_i^* used in the proof of the lemma 5, we just indicate the differences between the two constructions.

- We use *Closable* instead of *Closable$_n$* in the definition of the automaton involved in the specification of $N[ch]$. By proposition 1, this set is computable. With this modification, the simulation of every firing of an abstract transition followed later by a cut step is sound and complete.
- For every subtree tr_j^0 such that $ch(j) = k \in I$ and t_j^0 is an interruptible abstract transition, the corresponding cut step can be safely anticipated at the beginning of the sequence since the transformed sequence is still enabled. So, the tokens produced by the cut step are added to the marking $m_0[ch]$.
- For every subtree tr_j^0 such that $ch(j) = k \in I$ and t_j^0 is a non interruptible abstract transition, then in the ordinary Petri net $N[ch]$, we add a token to $p_{t_j^0,k}$ in $m_0[ch]$. Let us recall that consuming a token in $p_{t_j^0,k}$ simulates a cut step τ_k of a tree created by the firing of t_j^0. Furthermore, since $\Upsilon_i[ch]$ requires that this place is empty, the simulating sequence must perform the cut step simulation.
- For every subtree tr_j^0 such that $ch(j) = t' \in K(t_j^0)$, a token is added to $p_{t_j^0,t'}$ in $m_0[ch]$. Furthermore, the transformation indicated in the figure 16 is applied to ensure that the first occurrence of t'^- precedes the first occurrence of t''^- for any $t'' \in K(t) \setminus \{t'\}$. The added place $ctrl_{t'}$ does not modify the specification of $\Upsilon_i[ch]$.
- For every subtree tr_j^0 such that $ch(j) = \top$ and every $t \in K(t_j^0)$, we delete in $N[ch]$ every pair of transitions t^-, t^+. Indeed, such

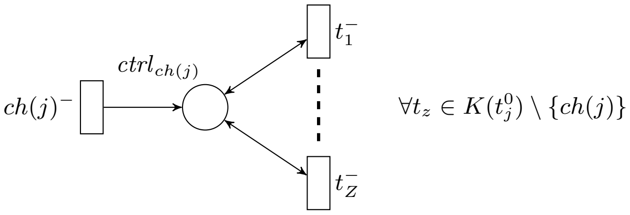


Fig. 16 Modification in $N[ch]$ for an interruptible transition t_j^0 such that $ch(j) \in K(t_j^0)$

Algorithm 3: Decide$\bot(N, tr_0, i)$

```
/* Is ⊥ reachable from $tr_0 \neq \bot$ by a cut step $\tau_i$, in $N$?                */
/* Let $(t_1^0, tr_1^0), \ldots, (t_{n_0}^0, tr_{n_0}^0)$ be the branches rooted in $v_0(tr_0)$   */
/* Let $Choices = (I \cup K(t_1^0) \cup \top) \times \cdots \times (I \cup K(t_{n_0}^0) \cup \top)$   */
foreach $ch \in Choices$ do
    foreach $j \in \{1, \ldots, n_0\}$ do
        if $ch(j) \in I \wedge \neg Decide\bot(N, tr_j^0, ch(j))$ then
            goto continue;
        end
    end
    Build $N[ch]$, $m_0[ch]$ and $\Upsilon_i[ch]$;
    if $ElemDecide(N[ch], m_0[ch], \Upsilon_i[ch])$ then
        return $true$;
    end
    continue:
end
return $false$;
```

transition t cannot occur in a firing sequence of N where the subtree tr_j^0 is present along the extended markings encountered by the sequence.

At last, assume that $tr_1 \neq \bot$. This case is handled by the general algorithm 4. This procedure looks for a sequence depending on the possible futures of every subtree rooted in $v_0(tr_0)$ and the possible past of every subtree rooted in $v_0(tr_1)$. The set $FBChoices$ represents every consistent choice (see fig. 12). Let $(fch, bch) \in FBChoices$ be such a choice, tr_j^0 be a subtree rooted in $v_0(tr_0)$ (with t_j^0 being the abstract transition labelling the edge to it) and $tr_{-j'}^1$ be a subtree rooted in $v_0(tr_1)$ (with $t_{-j'}^1$ being the abstract transition labelling the edge to it) then:

- $fch(j) = k \in I$ means that tr_j^0 has disappeared tr_1 by the firing of a cut step τ_k.

- $fch(j) = t' \in K(t_j^0)$ means that tr_j^0 has disappeared in tr_1 by the firing of an elementary transition denoted t'.
- $fch(j) = -k$ means that tr_j^0 has become tr_{-k}^1 in tr_1.
- $bch(-j') = m \in B^Q$ means that $tr_{-j'}^1$ has been created by the firing of $t_{-j'}^1$ when the current submarking over Q was m.
- $bch(-j') = k$ means that tr_k^0 in tr_0 has become $tr_{-j'}^1$.

Furthermore, we require that: $\forall j \in \{1, \ldots, n_0\}, t_j^0 = t_{fch(j)}^1 \wedge \forall k \in \{-1, \ldots, -n_1\}, fch(j) = k \Leftrightarrow bch(k) = j$ in order that the past and the future are consistent.

We explain now a round of the outer loop corresponding to some $(fch, bch) \in FBChoices$. First, it checks for all j such that $fch(j) \in I$ whether starting from tr_j^0, one can reach $\perp$ by a sequence ended by a cut step $\tau_{ch(j)}$. This is performed by a call to Decide$\perp$. Second, it checks for every j such that $fch(j) = -k$ whether starting from tr_j^0, one can reach tr_{-k}^1 by a recursive call where the depth of the extended markings has been decreased by 1. Third, it checks for every $-j'$ such that $bch(-j') = m \in B^Q$ whether starting from a subtree created by the firing of the abstract transition $t_{-j'}^1$ when the submarking over Q is m, one can reach $tr_{-j'}^1$ by a recursive call where the depth of the extended markings has been decreased.

When all answers are positive, it builds an ordinary Petri net $N[fch, bch]$, an initial marking $m_0[fch, bch]$ and a final semilinear set of markings $M_1[fch, bch]$ such that:

$$\boxed{\begin{array}{c}\exists \sigma \text{ s.t. } tr_0 \xrightarrow{\sigma}_N tr_1 \text{ and the induced behaviour} \\ \text{of the subtrees rooted in } v_0(tr_0) \text{ is defined by } (fch, bch) \\ \text{iff } \exists \sigma^*, \exists m_1 \in M_1[fch, bch] \text{ s.t. } m_0[fch, bch] \xrightarrow{\sigma^*}_{N[fch,bch]} m_1\end{array}}$$

Since $N[fch, bch]$ and $m_0[fch, bch]$ are similar to the net N^*, m_0^* used in the proof of the lemma 5, we just indicate the differences between the two constructions.

- We use *Closable* instead of *Closable*$_n$ in the definition of the automaton involved in the specification of $N[fch, bch]$.
- For every subtree tr_j^0 such that $fch(j) = k \in I$ and t_j^0 is an interruptible abstract transition, the corresponding cut step can be safely anticipated at the beginning of the sequence since the transformed sequence is still enabled. So, the tokens produced by the cut step are added to the marking $m_0[fch, bch]$.
- For every subtree tr_j^0 such that $fch(j) = k \in I$ and t_j^0 is a non interruptible abstract transition, then in the ordinary Petri net $N[fch, bch]$, we add a token to $p_{t_j^0,k}$ in $m_0[fch, bch]$.

- For every subtree tr^0_j such that $fch(j) = t' \in K(t^0_j)$, a token is added to $p_{t^0_j,t'}$ in $m_0[fch, bch]$. Furthermore, the transformation indicated in the figure 16 is applied to ensure that the first occurrence of t'^- precedes the first occurrence of t''^- for any $t'' \in K(t) \setminus \{t'\}$.
- For every subtree tr^0_j such that $fch(j) = -j'$ and every $t \in K(t^0_j)$, we delete in $N[fch, bch]$ every pair of transitions t^-, t^+.
- For every subtree tr^1_{-j} such that $bch(j) = m \in B^Q$, in $N[fch, bch]$ we add a transition t_{-j} which has the same input and output as ${t^1_{-j}}^-$ except that, in the automaton associated with $N[fch, bch]$, it additionally requires to be enabled that the submarking on Q is m. In order to ensure that this transition is fired exactly one, we add to it an input place $before_{t_{-j}}$ with $m_0[fch, bch](before_{t_{-j}}) = 1$ that will occur in the specification of $M_1[fch, bch]$.

It remains to define the semilinear set $M_1[fch, bch]$. Let $m_1 \in M_1[fch, bch]$. For every place $p \in \overline{Q}, m_1(p) = M(v_0(tr_1))(p)$. In the subnet corresponding to the automaton, exactly one state of the automaton $(m, (M_{tb})_{tb \in T_{ia}}, Tc)$ (i.e. one place) must be marked with the requirement that $m = M(v_0(tr_1))_{|Q}$. In m_1, the marking of control places defined $p_t, \overline{p}_t, p_{t',i}, p_{t',t}$, $ctrl_t$ and $before_{t'}$ must be the following one: $\forall t, t', i, m_1(p_t) = 0, m_1(\overline{p}_t) = 1, m_1(p_{t',i}) = 0, m_1(p_{t',t}) = 0, m_1(ctrl_t) \geq 1$ and $m_1(before_{t'}) = 0$.

The proofs of the two equivalences indicated in a frame are performed exactly as in the proof of lemma 5. □

We now focus on boundedness and finiteness of marked RPNs. The boundedness property ensures that there is a bound for any place of the ordinary markings labelling the nodes of any reachable extended marking and the finiteness property states that the number of reachable extended markings is finite. In Petri nets, these two properties are equivalent and decidable space exponentially in the size of the net [22]. In RPNs, the equivalence does not hold but decidability remains for both properties. However, as we use in the decision procedure a reachability test [19] for some Petri nets, our procedure is no more primitive recursive.

Theorem 4. *The boundedness problem is decidable for marked RPNs.*

Proof. Let us assume that some place p is unbounded, then for any integer n there is a reachable extended marking and one of its node for which the marking of p is greater than n. One can notice that the number of initial markings of nodes is finite (the initial markings of nodes composing the initial extended marking and the initial

Algorithm 4: Decide(N, tr_0, tr_1)

```
/* Is $tr_1$ reachable from $tr_0$, in $N$?                                  */
if $tr_0 == \perp$ then return $tr_1 == \perp$;
if $tr_1 == \perp$ then
    foreach $i \in I$ do
        if $Decide\perp(N, tr_0, i)$ then return $true$;
    end
    return $false$;
end
/* Let $(t^0_1, tr^0_1), \ldots, (t^0_{n_0}, tr^0_{n_0})$ be the branches rooted in $v_0(tr_0)$     */
/* and $(t^1_{-1}, tr^1_{-1}), \ldots, (t^1_{-n_1}, tr^1_{-n_1})$ the branches rooted in $v_0(tr_1)$  */
/* Let $FChoices =$
   $(I \cup K(t^0_1) \cup \{-1, \ldots, -n_1\}) \times \cdots \times (I \cup K(t^0_{n_0}) \cup \{-1, \ldots, -n_1\})$      */
/* Let $BChoices = (B^Q \cup \{1, \ldots, n_0\}) \times \cdots \times (B^Q \cup \{1, \ldots, n_0\})$          */
/* Let $FBChoices = \{(fch, bch) \in FChoices \times BChoices \mid \forall j \in$
   $\{1, \ldots, n_0\}, t^0_j = t^1_{fch(j)} \wedge \forall k \in \{-1, \ldots, -n_1\}, fch(j) = k \Leftrightarrow bch(k) = j\}$
   */
foreach $(fch, bch) \in FBChoices$ do
    foreach $j \in \{1, \ldots, n_0\}$ do
        if $fch(j) \in I \wedge \neg Decide\perp(N, tr^0_j, fch(j))$ then goto continue;
        if $fch(j) \in \{-1, \ldots, -n_1\} \wedge \neg Decide(N, tr^0_j, tr^1_{fch(j)})$ then
            goto continue;
        end
    end
    foreach $j \in \{-1, \ldots, -n_1\}$ do
        if $bch(j) \in B^Q \wedge \neg Decide(N, \lceil \Omega(t^1_j, bch(j)) \rceil, tr^1_j)$ then
            goto continue;
        end
    end
    Build $N[fch, bch]$, $m_0[fch, bch]$ and $M_1[fch, bch]$;
    if $ElemDecide(N[fch, bch], m_0[fch, bch], M_1[fch, bch])$ then
        return $true$;
    end
    continue:
end
return $false$;
```

markings associated with abstract transitions). So the place p is unbounded in the root of some marked RPN with the same structure as the original one and an initial extended marking which may be: either a simple node labelled by some $\Omega(t, m)$ where t is enabled in a node v of a reachable extended marking with current submarking on Q, $M(v)_{|Q}$, is equal to m, or some subtree of the initial extended marking. We note $\mathcal{T}r$ this subset of extended markings.

So the first step of the procedure consists, given a marked RPN, an abstract transition t^* and a submarking m^* on Q, to decide whether t^* will be enabled in some node v of a reachable extended marking such that $M(v)_{|Q} = m^*$. We reduce this problem to a reachability one where the final extended marking is $\perp$ and the RPN is transformed as follows. We add a control place *Ctrl*, marked in the root of the initial extended marking and unmarked in the other nodes. Then we restrict every Υ_i with the additional constraint $m(\mathit{Ctrl}) = 0$. This intermediate RPN has the same behaviour as the original one except that the cut steps are impossible in the root of the reachable extended markings. We add a second place *Reach* initially unmarked in every node. Afterwards, the set of indexes I is completed with a new index i^* such that $\Upsilon_{i^*} = \{m \mid (m_{|Q} = m^* \wedge t \text{ is enabled in } m) \vee m(\mathit{Reach}) = 1\}$. Finally, $\forall t \in T_{ab}, W^+(\mathit{Reach}, t, i^*) = 1$ and these are the only arcs connected to *Reach*. This additional index witnesses that the required condition is reached in a node and, due to the previous definitions, allows a sequence of cut steps from this node to the root leading to $\perp$.

When $\mathcal{T}r$ is computed, we decide whether a place p is unbounded in the root of some extended marking of $\mathcal{T}r$. We first compute for each immediate subtree of this extended marking the subset of indexes I' such that $i \in I'$ iff a cut step τ_i is enabled in the root of this subtree. This computation relies on the reachability procedure with the final extended marking being $\perp$. Furthermore, the marked net as well as the set of final markings are transformed in order that the cut steps enabled in the root are restricted to τ_i whereas in any other node the cut steps are unchanged using a similar construction as the one explained above.

Given a subtree of this extended marking, any firing sequence either closes this subtree by a cut step τ_i with $i \in I'$, either closes it by the firing of an elementary transition in the root of the current extended marking, or does not close it. Hence, we search for an infinite sequence increasing the marking of p in the root distinguishing the (finite) possible behaviours of sequence w.r.t. the subtrees of the initial extended marking. Each possible behaviour leads to a similar simulation given in the proof of proposition 1 by an ordinary Petri net. Given such a subtree, if we search for a sequence which does not close it, we ignore this subtree in the simulation. If we search for a sequence where a cut step closes it, we check whether $\perp$ is reachable from the subtree. If so, we add a transition in the ordinary net which fires once and whose effect is the one of the cut step at the root level. The case of an interrupt by an elementary transition is handled similarly.

□

Theorem 5. *The finiteness problem is decidable for marked RPNs.*

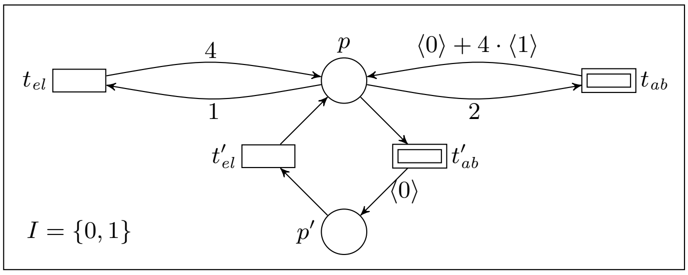
Proof. The reachability set of a marked RPN is infinite iff either the marked RPN is unbounded or it is bounded and the width or the depth of reachable extended markings are unbounded. From the previous proposition, we have just to decide for a bounded marked RPN whether the width or the depth are unbounded.

First, we deal with the case of bounded marked RPN with unbounded width. Such a case means that: $\exists t^* \in T_{ab}, \exists m^* \in B^Q$ such that $\forall n \in \mathbb{N}, \exists tr_n$ a reachable extended marking obtained by a sequence σ and with a node v whose at least n immediate descendants have been created by the firings of t^* when m^* was the current submarking on Q. Assume that there is a place p such that $W^-(p, t^*) > 0$ and consider tr'_n the reachable extended marking reached by a subsequence σ' obtained from σ by deleting the firings of t^* in v corresponding to the creation of these descendants and the subsequent firings in their subtrees. In tr'_n, $M(v)(p) \geq n$ which means that p is unbounded contrary to the hypothesis. Hence, such a t^* has no input place; and if t^* may be fired once, it may fired indefinitely. In order to check for such a case, we apply the procedure described for the boundedness problem.

The case of a bounded marked RPN with bounded width and unbounded depth is decided as follows. We call a *fresh* node, a node which was not present in the initial extended marking. We build a reachability graph until either we finish the building or we find an extended marking tr such that there are two fresh nodes v_1 and v_2 of tr issued by the firings of the same abstract transition t and with the same submarking on Q, m; and v_1 is an ancestor of v_2. As the marked RPN is bounded and the width is also bounded, this construction will terminate. The termination in the second case is equivalent to the unboundedness of the depth. Indeed, let σ be a firing sequence leading to tr and call σ_1 the subsequence obtained from σ by deleting all steps not originating from a descendant node of v_1. Then a similar sequence σ_2 can be played from tr at the level of v_2 leading to tr'. In tr', a new node v_3 descendant of v_2 has been created issued by the firings of the same abstract transition t and with the same submarking on Q, m. Iterating this process, we obtain a sequence of reachable extended markings with unbounded depth. □

Exploiting the linear invariants: In this subsection, we show how the linear invariant computation can be conducted in order to structurally analyse an RPN.

First, we define the incidence matrix W of an RPN. The rows of this matrix are composed of the places and of the abstract transitions.



(a) a RPN (only data relevant for matrix W are given)

$$\begin{array}{c|cc|cc|cccc|l}
 & t_{el} & t'_{el} & t_{ab} & t'_{ab} & (t_{ab},0) & (t_{ab},1) & (t'_{ab},0) & (t'_{ab},1) & \\ \hline
p & 3 & 1 & -2 & -1 & 1 & 4 & 0 & 0 & \rbrace P \\
p' & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & \\ \hline
t_{ab} & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & \rbrace T_{ab} \\
t'_{ab} & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & \\ \hline
 & \multicolumn{2}{c}{\underbrace{\qquad}_{T_{el}}} & \multicolumn{2}{c}{\underbrace{\qquad}_{T_{ab}}} & \multicolumn{4}{c}{\underbrace{\qquad\qquad\qquad}_{T_{ab}\times I}} &
\end{array}$$

(b) its matrix W

Fig. 17 a sample net and the corresponding matrix W

The intended meaning of a variable indexed by a place is its number of tokens while the interpretation of a variable indexed by an abstract transition is the current number of the subtrees created by its firing.

The columns of this matrix are composed of the transitions and the pairs (t,i) for t an abstract transition and $i \in I$. A column indexed by a transition represents its firing while a pair (t,i) represents a τ_i step in a subtree created by a firing of t.

Then W is defined by:

- $\forall p \in P, \forall t \in T_{el}, \forall t' \in T_{ab}$,
  $W(p,t) = W^+(p,t) - W^-(p,t)$ and $W(t',t) = 0$
- $\forall p \in P, \forall t, t' \in T_{ab}$ with $t' \neq t$,
  $W(p,t) = -W^-(p,t)$ and $W(t,t) = 1$ and $W(t',t) = 0$
- $\forall p \in P, \forall t, t' \in T_{ab}, \forall i \in I$ with $t' \neq t$,
  $W(p,(t,i)) = W^+(p,t,i)$ and $W(t,(t,i)) = -1$ and
  $W(t',(t,i)) = 0$

Figure 17 illustrates the definition of matrix W. The matrix is divided into 6 blocks depending on the type of rows and columns. Let us have a look at some items of the row indexed by place p: elementary transition t_{el} picks 1 token from p and puts 4 into it, thus the corresponding item of the matrix is 3; firing abstract transition t_{ab} consumes 2 tokens thus the corresponding item is -2 and the cut step associated with t_{ab} and index 0 (resp. 1) produces 1 token (resp. 4 tokens) thus the corresponding item is 1 (resp. 4). Let us have a look at the row indexed by abstract transition t_{ab}: firing t_{ab} creates one more subtree initiated by t_{ab} thus the corresponding item is 1 while firing any of the two cut steps corresponding to t_{ab} deletes one such subtree yielding an item -1.

Given a node v in an extended marking tr, we denote by $fire(v)^{tr}$ the vector, indexed on T_{ab}, such that, $\forall t \in T_{ab}$, $fire(v)^{tr}(t)$ is the number of subtrees issued from v by the firing of t. The next proposition justifies the choice of W for introducing invariants.

Proposition 2. *Let N be an RPN and $tr\xrightarrow{\sigma}tr'$ such that a node v is present in tr and tr' (and then in all intermediate extended markings along σ), let x be a solution of $x \cdot W = 0$ then:*

$$x \cdot (m(v)^{tr'}, fire(v)^{tr'}) = x \cdot (m(v)^{tr}, fire(v)^{tr}) \qquad (1)$$

The proof is obtained by an examination of the different kinds of steps.

For instance, assume that $\sigma = \tau_i$ is a cut step corresponding to abstraction transition t fired in a direct successor of v.
$fire(v)^{tr'}(t) = fire(v)^{tr}(t) - 1$, $\forall t' \neq t, fire(v)^{tr'}(t') = fire(v)^{tr}(t')$ and $\forall p \in P, m(v)^{tr'}(p) = m(v)^{tr'}(p) + W^{+}(p,t,i)$.
Thus $(m(v)^{tr'}, fire(v)^{tr'}) - (m(v)^{tr}, fire(v)^{tr})$ is the column of W indexed by (t,i). Hence $x \cdot ((m(v)^{tr'}, fire(v)^{tr'}) - (m(v)^{tr}, fire(v)^{tr})) = 0$ as required.

As in ordinary Petri nets, when tr is the initial extended marking, the equation (1) is called a linear invariant.

In order to obtain linear invariants, one can compute a generative family of solutions $\{x_1, \ldots, x_n\}$ of this equation. We obtain a superset of the reachable "states" of a node in a tree by the set of equations $\forall 1 \leq i \leq n, x_i \cdot (m(v)^{tr}, fire(v)^{tr}) = x_i \cdot (m(v)^{tr_0}, fire(v)^{tr_0})$. The same overestimation can be done for the reachable ordinary marking space of a node v dynamically created by the firing of t when the current submarking on Q is m^*: $\forall 1 \leq i \leq n, x_i \cdot (m(v)^{tr}, fire(v)^{tr}) = x_i \cdot (\Omega(t, m^*), \mathbf{0})$ (where $\mathbf{0}$ denotes the null vector).

Compared to the Petri nets model, we have here additional sources of overestimation: the matrix W^*, the interrupt effect of the elementary transitions (specified by the mapping K) which are not taken into account and finally the fact that depending on the initial marking $\Omega(t, m)$ some transitions are dead and thus may be excluded from the

Algorithm 5: *structReach*

```
input  : an ordinary marking $m$
output: a set of invariants and a set of transitions
$T_{live} = \emptyset$;
$New = \emptyset$;
$In = \emptyset$;
repeat
    $New = \emptyset$;
    $In = Invariant(N, m, T_{live})$;
    foreach $t \in T \setminus T_{live}$ do
        Build a linear problem $Pb$ in $\mathbb{N}^P$ with $In, W^-(P,t)$ and $W^*(Q,t)$;
        if $Pb$ admits a solution then
            $New = New \cup \{t\}$;
        end
    end
    $T_{live} = T_{live} \cup New$;
until ($New == \emptyset$) ;
return $\langle In, T_{live} \rangle$;
```

computation. Since this last factor often happens in practical cases, we describe now an iterative method which tackles this problem. In fact, the method is also applicable to Petri nets but it is of limited interest in this case since usually the transitions of a Petri net are not dead.

The algorithm 5 simultaneously computes a set of linear invariants fulfilled by markings reachable from m, an ordinary marking, and a superset of the transitions enabled at least once from m.

More precisely, it initialises T_{live} as the empty set. Then it computes the positive invariants for the recursive Petri net whose transitions are reduced to T_{live}. In the algorithm 5, the function *Invariant* returns a generative family of invariants (see [2] for efficient computation of such families).

For each transition not belonging to T_{live}, it builds a linear problem with the invariants and the firing conditions of this transition. If this problem admits a solution, it is possibly enabled and so, it adds it to T_{live}. This resolution may be performed for rational numbers with a polynomial time complexity or for integer numbers with a higher complexity. However, it may occur that the problem admits a rational solution but no integer one. In the Petri net context, experiments show that this situation is rarely encountered and yield to select the rational resolution. This process is iterated until T_{live} is saturated. The returned invariants specify an overestimation of reachable markings with their current firings of abstract transitions.

We finally describe how the linear invariants can be used to obtain information about the tree structure of the reachable extended markings.

We build a graph whose nodes are the tuples (t, m) with $t \in T_{ab}$ and $m \in B^Q$, compatible with t. There is an edge from (t, m) to (t', m') if starting from the marking $\Omega(t, m)$ a thread may fire t' when the current submarking on Q is m'. In order to determine such an edge, we compute the invariants associated with $\Omega(t, m)$ by a call to *structReach*. Then we build a linear problem including these invariants, the firing condition of t' and fixing the submarking on Q to be m'. If such a problem has a solution then an edge is added.

This graph is a skeleton for the dynamical structure of the extended markings. For instance, if it is acyclic, then any reachable extended marking has a bounded depth.

4 Conclusion

We have introduced the model of recursive Petri nets for which we have developed some theory. On the one hand, we have studied its expressive power showing its ability to model complex mechanisms of DESs like interrupts, fault-tolerance, remote procedure calls and environment-driven behaviours. We have also proved that some of these patterns cannot be modelled by ordinary Petri nets. On the other hand, we have designed decidability algorithms of some problems: reachability, finiteness and bisimulation. At last, we have developed the concept of linear invariants for this kind of nets and designed associated efficient computations.

Our goal is the experimentation of this model for industrial case studies. In order to achieve it, we plan to implement our algorithms in a verification tool managing extensions of Petri nets like [25,17]. From a theoretical point of view, we now want to study more thoroughly the relations between the different extensions of Petri nets. For instance, the exact relation between parallel rewrite systems [20] and recursive Petri nets is still an open problem.

Acknowledgments. We thank the anonymous referees for their helpful comments.

References

1. C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, September 1999.

2. J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. a comparative study of algorithms for computation of minimal P-semiflows. In *Advances in Petri Nets*, volume 483 of *Lecture Notes Computer Science*, pages 79–112. Springer-Verlag, June 1990.
3. C. Dufourd, A. Finkel, and P. Schnoebelen. Reset nets between decidability and undecidability. In *Proc of the* 25^{th} *Int. Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.
4. S. Eilenberg and M. P. Schützenberger. Rational sets in commutative monoïds. *Journal of Algebra*, 13:173–191, 1969.
5. J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the European Association for Theoretical Computer Science*, 52:245–262, 1994.
6. S. Haddad and D. Poitrenaud. Decidability and undecidability results for recursive Petri nets. Technical Report 019, LIP6, Paris VI University, Paris, France, September 1999.
7. S. Haddad and D. Poitrenaud. Theoretical aspects of recursive Petri nets. In *Proc. of the* 20^{th} *Int. Conf. on Applications and Theory of Petri nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 228–247, Williamsburg, VA, USA, June 1999. Springer-Verlag.
8. S. Haddad and D. Poitrenaud. Modelling and analyzing systems with recursive Petri nets. In *Proc. of the* 5^{th} *Workshop on Discrete Event Systems - Analysis and Control*, pages 449–458, Gand, Belgique, August 2000. Kluwer Academics Publishers.
9. S. Haddad and D. Poitrenaud. Checking linear temporal formulas on sequential recursive Petri nets. In *Proc of the* 8^{th} *International Symposium on Temporal Representation and Reasonning*, pages 198–205, Cividale del Friuli, Italie, June 2001. IEEE Computer Society Press.
10. M. Jantzen. On the hierarchy of Petri net languages. *RAIRO*, 13(1):19–30, 1979.
11. P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148:281–301, 1995.
12. P. Jančar, J. Esparza, and F. Moller. Petri nets and regular processes. *Journal of Computer and System Sciences*, 59(3):476–503, 1999.
13. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts.* Monographs in Theoretical Computer Science. Springer-Verlag, 1997.
14. A. Kiehn. Petri nets systems and their closure properties. In *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 306–328. Springer-Verlag, 1989.
15. M. Köler and H. Rölke. Properties of object Petri nets. In *Proc. of the* 25^{th} *Int. Conf. on Application and Theory of Petri Nets*, volume 3099 of *Lecture Notes Computer Science*, pages 278–297, Bologna, Italy, June 2004. Springer-Verlag.
16. O. Kouchnarenko and Ph. Schnoebelen. A model for recursive-parallel programs. In *Proc. of the* 1^{st} *Int. Workshop on Verification of Infinite State Systems*, volume 5 of *Electronic Notes in Theor. Comp. Sci.*, Pisa, Italy, August 1997. Elsevier Science.
17. O. Kummer, F. Wienberg, M. Duvigneau, J. Schumacher, M. Köler, D. Moldt, H. Rölke, and R. Valk. An extensible editor and simulation engine for Petri

nets: RENEW. In *Proc. of the* 25th *Int. Conf. on Application and Theory of Petri Nets*, volume 3099 of *Lecture Notes Computer Science*, pages 484–493, Bologna, Italy, June 2004. Springer-Verlag.

18. I. Lomazova and Ph. Schnoebelen. Some decidability results for nested Petri nets. In *Proc. of the* 3rd *Int. Andrei Ershov Memorial Conf. Perspectives of System Informatics*, volume 1755 of *Lecture Notes Computer Science*, pages 208–220, Novosibirsk, Russia, July 2000. Springer-Verlag.
19. E.W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. of the* 13th *Annual Symposium on Theory of Computing*, pages 238–246, 1981.
20. R. Mayr. Combining Petri nets and PA-processes. In *Proc. of the* 3rd *Int. Symposium on Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 547–561, Sendai, Japan, 1997. Springer-Verlag.
21. R. Mayr. Process Rewrite Systems. *Information and Computation*, 156(1):264–286, 2000.
22. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.
23. C. Reutenauer. *The Mathematics of Petri Nets.* New York, USA: Prentice-Hall, 1990.
24. A. El Fallah Seghrouchni and S. Haddad. A recursive model for distributed planning. In *Proc. of the Second Int. Conf. on Multi-Agent Systems*, pages 307–314, Kyoto, Japon, December 1996.
25. C. Sibertin-Blanc. CoOperative Objects: Principles, Use and Implementation. In *Concurrent Object-Oriented Programming and Petri Nets, Advances in Petri Nets*, volume 2001 of *Lecture Notes Computer Science*, pages 216–246. Springer-Verlag, 2001.
26. R. Valk. On the computational power of extended Petri nets. In *Proc. of the* 7th *Int. Symposium on Mathematical Foundations of Computer Science*, volume 64 of *Lecture Notes Computer Science*, pages 526–535, Zakopane, Poland, September 1978. Springer-Verlag.
27. R. Valk. Self-modifying nets, a natural extension of Petri nets. In *Proc. of the* 5th *Int. Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes Computer Science*, pages 464–476, Udine, Italy, July 1978. Springer-Verlag.
28. R. Valk. Petri nets as token objects: An introduction to elementary object nets. In *Proc. of the* 19th *Int. Conf. on Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes Computer Science*, pages 1–25. Springer-Verlag, June 1998.

A Recall of bisimulation definitions and results

Since the bisimulation relation is defined on transitions systems, we recall their definition. For instance, the state graphs of a Petri net and of an RPN can be viewed as transitions systems.

Definition 16. *A labelled transition system is a tuple $\langle \Sigma, S, \{\xrightarrow{a}\}_{a\in\Sigma}\rangle$ where*

- *Σ is a finite alphabet*
- *S is a set of states*
- *$\forall a \in \Sigma, \xrightarrow{a}$ is a binary relation included in $S \times S$*

Below, we define the bisimulation relation $\sim$ and weaker relations than the bisimulation $\{\sim_n\}_{n\in\mathbb{N}}$.

Definition 17. *Let LTS and LTS' be two labelled transition systems defined by $LTS = \langle \Sigma, S, \{\xrightarrow{a}\}_{a\in\Sigma}\rangle$ and $LTS' = \langle \Sigma, S', \{\xrightarrow{a}\}_{a\in\Sigma}\rangle$, then:*

- *$R \subseteq S \times S'$ is a bisimulation relation iff $\forall s \in S$, $\forall s' \in S'$ s.t. $s\ R\ s'$, $\forall a \in \Sigma$:*
  1. *$\forall s_1 \in S, s\xrightarrow{a}_{LTS} s_1 \Rightarrow \exists s'_1 \in S'$ s.t. $(s'\xrightarrow{a}_{LTS'} s'_1) \wedge (s_1\ R\ s'_1)$*
  2. *$\forall s'_1 \in S', s'\xrightarrow{a}_{LTS'} s'_1 \Rightarrow \exists s_1 \in S$ s.t. $(s\xrightarrow{a}_{LTS} s_1) \wedge (s_1\ R\ s'_1)$*

Two states s and s' are bisimilar iff there exists some bisimulation relation R such that $s\ R\ s'$. This is denoted by $(LTS, s) \sim (LTS', s')$.

- *$\forall s \in S, \forall s' \in S', (LTS, s) \sim_0 (LTS', s')$*
- *$\forall s \in S, \forall s' \in S', (LTS, s) \sim_{n+1} (LTS', s')$ iff for every $a \in \Sigma$:*
  1. *$\forall s_1 \in S, s\xrightarrow{a}_{LTS} s_1 \Rightarrow \exists s'_1 \in S'$ s.t.*
     $(s'\xrightarrow{a}_{LTS'} s'_1) \wedge (LTS, s_1) \sim_n (LTS', s'_1)$
  2. *$\forall s'_1 \in S', s'\xrightarrow{a}_{LTS'} s'_1 \Rightarrow \exists s_1 \in S$ s.t.*
     $(s\xrightarrow{a}_{LTS} s_1) \wedge (LTS, s_1) \sim_n (LTS', s'_1)$

We may explain the relation $(LTS, s) \sim_n (LTS', s')$ as follows: first, one builds the labelled transition system LTS^s_n (resp. $LTS'^{s'}_n$) by considering the behaviour tree of LTS (resp. LTS') issued from s (resp. s') up to n steps. Then $(LTS, s) \sim_n (LTS', s')$ iff $(LTS^s_n, s) \sim (LTS'^{s'}_n, s')$. We now introduce two useful notations for the next lemma.

Definition 18. *Let $LTS = \langle \Sigma, S, \{\xrightarrow{a}\}_{a\in\Sigma}\rangle$ be a labelled transition system,*

- *Inc^{LTS}_n is the family of initialised systems incompatible with LTS for $\sim_n$:*

$$Inc^{LTS}_n = \{(LTS', s') \mid \forall s \in S, (LTS, s) \not\sim_n (LTS', s')\}$$

- $\xrightarrow{*}$ *is the reflexive and transitive closure of the union of* $\xrightarrow{a}$. *In other words,* $s \xrightarrow{*}_{LTS} s'$ *iff* s' *is reachable from* s.

This lemma establishes a characterisation of the bisimulation of a finite transition system by a (possibly infinite) transition system.

Lemma 6. *[12] Let* $LTS = \langle \Sigma, S, \{\xrightarrow{a}\}_{a \in \Sigma} \rangle$ *be a finite labelled transition system (with* $n = |S|$*) and* $LTS' = \langle \Sigma, S', \{\xrightarrow{a}\}_{a \in \Sigma} \rangle$ *be a (possibly infinite) labelled transition system, then* $\forall s \in S, \forall s' \in S'$,

$$(LTS, s) \sim (LTS', s') \Leftrightarrow \begin{cases} (LTS, s) \sim_n (LTS', s') \wedge \\ \nexists\ (LTS', s'') \in Inc_n^{LTS}\ \ s.t.\ s' \xrightarrow{*}_{LTS'} s'' \end{cases}$$