Chapter 4
Decidability and complexity of Petri net problems

Serge HADDAD[1]

1. Introduction

In the previous chapter we have emphasised that, on the one hand, the efficiency of methods based on the reachability graph strongly depends on its size and, on the other hand, that these methods are only applicable to bounded nets. So in this chapter, we will evaluate the complexity of the reachability graph and we will design verification methods applicable when the graph is infinite.

We begin by a summary of the main notions of decidability and complexity that will be used in the other sections. Then we recall negative results about the reachability graph both w.r.t. decidability when the graph is infinite and w.r.t. complexity in the finite case.

Afterwards we develop usual methods for the case of infinite graphs. The first method, called the construction of covering graph, is an adaptation of the construction of the reachability graph. This algorithm substitutes an infinite value (denoted ω) to the marking of a place p when it detects the possibility to reach markings greater than the current marking with arbitrary great values in p. This finite graph is an abstraction of the reachability graph which allows to decide some generic properties. The second method relies on the computation of a bound of the length of shortest paths witnessing some property. Then

[1]LSV, ENS Cachan, 61, avenue du Président Wilson 94235 Cachan Cedex France, (haddad@lsv.ens-cachan.fr)

it is sufficient to look for the existence of a path with a bounded length in order to check whether the property holds. The third method starting from a target marking, performs backward firings from a set of markings in order to reach the initial marking. The underlying procedure ensures the termination of the algorithm. We illustrate both methods on the covering property (i.e. the reachability of a marking greater or equal than a given marking).

None of the previous methods solves the reachability problem: given two markings of a net, decide whether the second marking is reachable from the first one. So we will informally describe an algorithm for the reachability problem. The importance of this algorithm is twofold. First it is a very involved algorithm and in addition it seems to be a limit result of decidability in Petri nets.

The last two sections are devoted to the expressive power of the Petri net model. Knowing that this model is less expressive than the model of Turing machines, it is tempting to introduce new control mechanisms and to adapt the verification methods to these extended models. We present three relevant extensions of Petri nets. The nets with inhibitor arcs allow to forbid the firing of a transition if the marking of a place is greater than some value. In a self-modifying net, a transition produces or consumes a number of tokens that depends on the current marking. A recursive net contains abstract transitions whose firing leads to a state consisting in a dynamical tree of marked nets. For each model, we point out the impact of the extension on the verification of properties.

A usual way to compare models w.r.t. their expressive power is the study of the languages generated by firing sequences. We end the chapter by a description of properties of Petri net languages and a comparison of this family of languages with the standard hierarchy of families of languages.

The interested reader will find in [ESP 94] a detailed panorama of decidability and complexity results in Petri nets. He can also refer to [ESP 98] which covers complementary features of those discussed here.

The notations of the previous chapter also apply here. For sake of readability we recall below some results of the previous chapter that we intensively use in the sequel.

Lemma 1 (Monotonicity lemma) *Let R be a Petri net,*

1. $\forall\, m_1 \leq m'_1\ m_1 \xrightarrow{\sigma} m_2 \Rightarrow\ m'_1 \xrightarrow{\sigma} m'_2$ *with* $m_2 \leq m'_2$
2. *Furthermore, let p be a place such that $m_1(p) < m'_1(p)$ then $m_2(p) < m'_2(p)$*

Lemma 2 (Koenig lemma) *Let A be a tree with finite degree (i.e. every vertex has a finite number of successors) and including an infinite number of vertices; then A admits an infinite branch.*

Lemma 3 (Extraction lemma) *Let $m_0, m_1, \ldots$ be an infinite sequence of vectors of $\mathbb{N}^{\{1,\ldots,k\}}$, this sequence admits a (largely) increasing subsequence.*

Proposition 4 (Characterisation of a bounded net) *Let R be a net:*

(R, m_0) *is unbounded iff*

(R, m_0) *admits a firing sequence* $m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} m_2$ *with* $m_1 < m_2$

2. Decidability and complexity notions

The summary given here is enough to understand the remainder of the chapter but cannot replace a deeper treatment of these topics.The interested reader can refer to specialised books (e.g. [PAP 94]).

Definition 5 (Notion of problem) *A problem is a partial function f from A to B. The items of A and B has at least one finite representation able to be read or written by a machine or a program.*
In case where B is the boolean domain, one says that it is a decision problem.

Informally an item a of A is an instance of the problem and $f(a)$ is the result of this problem for this instance. Here are two simple examples:

- The construction of the reachability graph is a problem where A is the set of Petri nets and B is the set of **finite** graphs. f is the **partial** function which associates with a **bounded** net its reachability graph.
- The liveness of a net is a problem where A is the set of Petri nets and B is the boolean domain. f is the **total** function which associates with a net a boolean witnessing whether the net is live.

To be more precise, we should indicate the chosen representation for the input and the result since in some cases this choice has an impact on the complexity of the problem (e.g. unary representation of integers). We will not do it assuming a reasonable and "standard" representation of the objects we study.

Our goal is to solve problems using programs. Following Church thesis, we do not precise the programming language assuming that it has the basic constructors: *if then else, while, for, ...*

Definition 6 (Recursive problem) *A problem f from A to B is recursive if there exists a program which takes as input an item $a \in Domain(f) \subseteq A$ and terminates and produces (or prints) $f(a)$.*
When B is the boolean domain, one says that the problem is decidable.

Once one knows that a problem is recursive, i.e. there exists an algorithm to solve it, we need to measure the efficiency of the computation. The main difficulty is the constructor *while*. Indeed even if one can prove that the program will eventually exit the loop, it is sometimes impossible to predict the number of times that the loop block will be executed. Let us call a *primitive program*, a program that has only constructors *for*, *if then else* and the *concatenation* and whose one variable is initialised with the size of the input and the other variables are initialised with zero.

Definition 7 (Primitive recursive problem) *A problem f is primitive recursive if there exists a primitive program which takes as input an item $a \in Domain(f) \subseteq A$, terminates and produces (or prints) $f(a)$.*

Informally on can say that a non primitive recursive problem can be solved but that its complexity is unknown. Thus one tries to design primitive programs or at least programs that we can transform later in primitive programs in order to measure their complexity. The two most often used complexity criteria are time complexity (number of instructions performed) and space complexity (size of the required memory). In the second case, the input and the output are supposed to be on a secondary memory and do not count in the measure.

Definition 8 *Let $Comp$ be a function from $\mathbb{N}$ to $\mathbb{R}$, f be a problem and prog be a program that solves f. We note $|a|$ the size of an object a, then:*
prog is time (resp. space) bounded by $Comp$ if for every $a \in Domain(f)$, prog performs (resp. uses) at most $Comp(|a|)$ instructions (resp. memory cells).

This very precise complexity measure depends on both the choice of the programming language and on the size of the memory cells. In fact, we rather need to characterise complexity classes related to family of functions as for instance:

- $\mathcal{P}$, the class of polynomial functions
- $\mathcal{EXP}$, the class of functions that can be written 2^{Comp} with $Comp \in \mathcal{P}$
- $\mathcal{LOG}$, the class of functions that can be written $log(Comp)$ with $Comp \in \mathcal{P}$

Thus we obtain a more robust notion of complexity measure.

Definition 9 (Complexity classes) *Let $\mathcal{C}$ be a class of functions from $\mathbb{N}$ to $\mathbb{R}$ and f be a problem:*

- *f belongs to $\mathcal{C}time$ or more simply to $\mathcal{C}$, if there exists a program whose execution time is bounded by $Comp$, a function from $\mathcal{C}$.*
- *f belongs to $\mathcal{C}space$, if there exists a program that solves f and whose memory size is bounded by $Comp$, a function from $\mathcal{C}$.*

Numerous decision problems can be solved using non deterministic choices (e.g. choice of a fireable transition) and to backtrack when the choice leads to a dead end (i.e. negative answer). Otherwise stated, the execution can be visualised as a tree where every branch represents an execution sequence and where the algorithm returns true if at least one execution returns true. Assume that one could use an imaginary machine which after every non deterministic choice can simultaneously executes the different alternatives, the time complexity would be equal to the execution time of the longest branch.

Formalising it, given a class $\mathcal{C}$ one denotes $\mathcal{NC}time$ or more simply $\mathcal{NC}$, the set of problems decidable by a program including non deterministic choices such that a function of C bounds the execution time of every execution. This notion is also applicable to the space complexity.

Remarks:

- Clearly a deterministic program can occupy at most a number of memory cells proportional to the number of performed instructions. Hence $\mathcal{C}time \subset \mathcal{C}space$
- Using Savitch procedure [AHO 74], every program can be transformed into a deterministic one that uses a memory size quadratic w.r.t. the memory size of the original program. Hence $\mathcal{NC}time \subset \mathcal{C}space$ as soon as $\mathcal{P} \subseteq \mathcal{C}$.
- A deterministic program is a particular non deterministic program. Hence $\mathcal{C}time \subset \mathcal{NC}time$

We summarise below the relations between the most usual complexity classes and that will be used in this book.

$$\mathcal{NLOG}space \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}space \subseteq \mathcal{EXP} \subseteq \mathcal{NEXP} \subseteq \mathcal{EXP}space$$

Some inclusions are strict (e.g. $\mathcal{P} \neq \mathcal{EXP}$ and $\mathcal{P}space \neq \mathcal{EXP}space$) while for some others the problem remains open (e.g. $\mathcal{P} \neq \mathcal{NP}$? and $\mathcal{NP} \neq \mathcal{P}space$?).

The complexity that we have described until now is focused on upper bounds for the complexity of a problem. The search for lower bounds is also interesting, since one the one hand it avoids a useless search of (too much) efficient algorithms, and on the other hand it allows to define a notion of optimality for some algorithms.

In order to formalise the concept of lower bound, one first defines the notion of reduction.

Definition 10 (Problem reduction) *Let f be a problem from A to B and g be a problem from A' to B. f reduces to g with complexity $\mathcal{C}time$ (resp. $\mathcal{C}space$) if there exists a program,*

- *which takes as input an object $a \in Domain(f)$ and provides a result $a' \in Domain(g)$,*
- *such that $g(a') = f(a)$,*
- *and is time (resp. space) bounded by a function of $\mathcal{C}$.*

The two main kinds of reductions are $\mathcal{P}time$ reductions (enough for classes including $\mathcal{P}time$) and $\mathcal{LOG}space$ reductions (for smaller classes than $\mathcal{P}time$).

Definition 11 (Hard and complete problems) *Let f be a problem,*

- *f is $\mathcal{C}$-hard if every problem of $\mathcal{C}$ is reducible to f.*
- *f is $\mathcal{C}$-complete if f is $\mathcal{C}$-hard and f belongs to $\mathcal{C}$.*

When a problem is $\mathcal{C}$-hard, it is useless to look for an algorithm which belongs to a smaller class than $\mathcal{C}$. A problem $\mathcal{C}$-complete is one of *"the most complex problems of the class $\mathcal{C}$"*. For instance, the reachability of a vertex starting from another one in a graph is a problem $\mathcal{NLOG}space$-complete and the simultaneous satisfaction of a set of disjunctive propositional clauses is a problem $\mathcal{NP}$-complete.

3. Theoretical results about the reachability graph

Given two Petri nets which have the same set of places, it is interesting to compare their reachability set. For instance, one wishes to compare the reachability set of a safe system to the one of a system that one wants to certify. Unfortunately, we have here the first negative results.

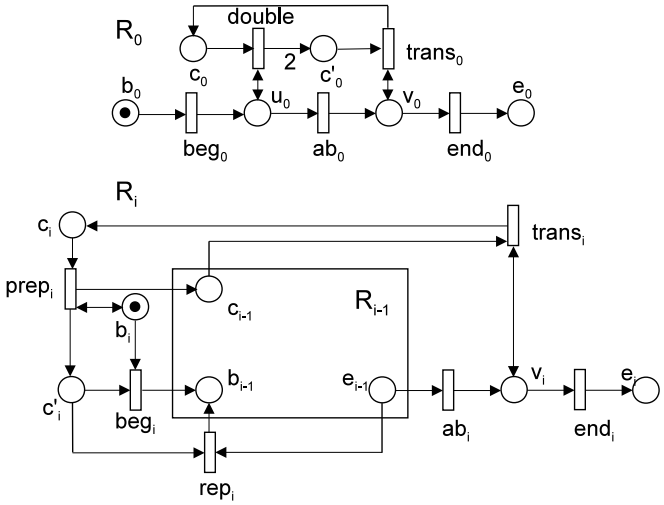

Figure 1: A family of bounded nets

Proposition 12 (Comparison of reachability sets [HAC 75]) *Given two Petri nets over the same set of places, the problems of equality and inclusion reachability sets are undecidable.*

When restricting these questions to bounded nets, they obviously become decidable. But what it is their complexity ? Before answering it let us examine the complexity of the construction of reachability graph for bounded nets.

Proposition 13 (Complexity of the reachability graph) *Given a* ***bounded*** *Petri net, the problem of the construction of the reachability graph (or set) is not primitive recursive.*

Proof
We show that there exists a family of marked bounded nets indexed by $\mathbb{N}$ such that the size of a marked net is proportional to its index and the size of its reachability graph is not a primitive recursive function of the index.

We first compute an upper bound $f_n(x)$ of the absolute value of a variable after the execution of a primitive program including n constructors and whose variables are initialised with absolute values less or equal than $|x|$. W.l.o.g.,

we consider the case $x \geq 2$. More precisely, the programming language has a basis instruction, the affectation of a sum of variables $x = y(+/-)z$ and three constructors: the choice *if then else* with the restriction that the test does not modify the value of variables, *the concatenation* and the loop *for* whose number of rounds is fixed by the value of a variable. Let us show that the functions f_n defined below are appropriate:

- $f_0(x) = 2.x$
- $f_{i+1}(x) = f_i^{(x)}(x)$ where $f_i^{(x)}$ denotes the composition x times repeated of f_i

Using an immediate recurrence, one checks that the functions f_i are strictly increasing and that $f_{i+1}(x) > f_i(x)$.

If a program does not include constructors, it has exactly a basis instruction and the value of the affected variable cannot exceed $2.x$. Assume that the upper bound has been proved for i constructors and let us study a program with $i+1$ constructors. If the more external constructor is the choice then the execution is one of its branches. Hence $f_i(x)$ bounds the content of variables. If the more external constructor is the concatenation then the execution is the one of a program with at most i constructors taking as inputs variables computed by a program with at most i constructors whose variables are initially less or equal than x w.r.t. the absolute value. Hence $f_i(f_i(x))$ bounds the content of variables. If the more external constructor is the loop *for* then there are at most x rounds and this program is equivalent to at most $x-1$ concatenations of programs with at most i constructors. Hence $f_i^{(x)}(x)$ bounds the content of variables.

Applying the usual diagonalisation technique and defining $f(x) = f_x(x)$, one proves that f is not a primitive recursive function since it cannot be bounded by any function f_n.

Let R_i be the family of nets described figure 1. Define $m_i = i.\overrightarrow{c_i} + \overrightarrow{b_i}$. One proves that (R_i, m_i) is bounded and that c_i can reach any value between 0 and $f(i)$. Since the size of the marked net is proportional to its index, this achieves the demonstration.

In order to obtain this result without entering the details, one establishes (for $i > 1$) by recurrence the following points:

- R_i has a P-semiflow $z_i = b_i + v_i + e_i + b_{i-1} + v_{i-1} + e_{i-1} + \cdots + b_0 + v_0 + u_0 + e_0$
- Given an initial marking m such that $z_i^t \cdot m = m(b_i) = 1$ the maximal sequences of R_i have the following pattern,
  $\sigma = prep_i^p.beg_i.\sigma_1.rep_i.\sigma_2.\ldots.rep_i.\sigma_q.ab_i.trans_i^r.end_i$ where:

- $p \leq m(c_i) + m(c'_i)$,
- $q \leq p$,
- σ_1 is a sequence of R_{i-1} beginning by marking m_1 fulfilling a condition like the one of m and such that:
  $m_1(c_{i-1}) + m_1(c'_{i-1}) + \cdots + m_1(c_0) + m_1(c'_0) =$
  $m(c_{i-1}) + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p$
- For $1 < q' \leq q$ σ'_q is a sequence of R_{i-1} beginning by a marking $m_{q'}$ fulfilling a condition like the one of m and such that:
  $m_{q'}(c_{i-1}) + m_{q'}(c'_{i-1}) + \cdots + m_{q'}(c_0) + m_{q'}(c'_0) \leq$
  $f_{i-1}^{(q'-1)}(m(c_{i-1}) + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p)$
- $r \leq f_{i-1}^{(q)}(m(c_{i-1} + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p)$
- for every m' reached from m during σ,
  $m'(c_i) + m'(c'_i) + \cdots + m_q(c_0) + m_q(c'_0) \leq$
  $f_i(m(c_i) + m(c'_i) + \cdots + m(c_0) + m(c'_0))$

• One exhibits a sequence σ that reaches any possible value of $m'(c_i)$.

$\diamondsuit$

This proposition means that every method based on the reachability graph construction has an unpredictable complexity. This justifies the interest of structural methods presented in the previous chapter. The following proposition states, that no other method can provide significantly better results.

Proposition 14 *[MAY 81] Given two* ***bounded*** *Petri nets over the same set of places, the problem of equality and inclusion of reachability sets are not primitive recursive.*

4. Analysis of unbounded Petri nets

Verification methods for unbounded Petri nets must either adapt or omit the construction of the reachability graph. Since we illustrate three such methods with the covering problem, let us recall its definition.

Definition 15 (Covering of a marking) *Let (R, m_0) be a Petri net and m_c be a marking. The net covers m_c if there exists $m \in A(R, m_0)$ such that $m \geq m_c$.*

This problem presents (at least) two interests. First, it has numerous applications. For instance, if one wants to check whether two places are in mutual

exclusion, one tries to cover the marking consisting of a token in every place. Furthermore, among the decision algorithms of generic properties, those relative to the covering problem are the simplest ones and can be more easily adapted to some extensions of Petri nets.

4.1. *Construction of the covering graph*

Let us look for a construction of the reachability graph that can detect (on the fly) that the net is unbounded and stop the construction. We build a tree whose every vertex is labelled by a marking as follows:

1. The root is labelled by the initial marking,
2. every *processed* vertex has for sons a set of vertices whose labels are the markings reachable by a transition firing from the marking of that vertex,
3. the processing of a new vertex is only planned if its marking is different from every marking which is present at the time of its creation,
4. the construction is aborted if on a branch that leads to a new vertex there is a vertex whose marking is strictly smaller than the marking of the new vertex,
5. if the construction terminates, one identifies the vertices with same markings which yield the reachability graph.

Lemmata 2 and 3 ensure that a unbounded will be detected by point 4 and proposition 4 ensures that one does not abort the construction of a bounded net.

We want to modify the previous procedure in order to check the covering of a marking by a net. This leads to the notion of covering tree $AC(R, m_0)$, proposed by Karp and Miller [KAR 69]. To this aim, one introduces a new set by adding to $\mathbb{N}$ a greatest element denoted ω : $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ and one extends the order $<$ and the operations $+$ and $-$ as follows:
$\forall n \in \mathbb{N} : n < \omega,\ n + \omega = \omega + n = \omega,\ \omega - n = \omega$ $\qquad$ ($n - \omega$ is not defined)
One similarly extends $+, -, <$ to $(\mathbb{N}_\omega)^P$ called the set of ω-markings. Given a ω-marking one calls a place with marking ω, an ω-component and otherwise a finite component.

The searched result is the following equivalence.

Proposition 16

(R, m_0) *covers* $m_c \Leftrightarrow \exists$ *a vertex of* $AC(R, m_0)$ *labelled by* m_ω *s.t.* $m_c \leq m_\omega$

If one examines an increasing repetitive sequence $m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} m_2$ with $m_1 < m_2$, it is clear that repeating the sequence, the marking of every place for which the markings m_1 et m_2 are different will infinitely increase. Let $m_{2,\omega}$ be the ω-marking, obtained by substituting in m_2 the marking of such places by ω; then every marking less than $m_{2,\omega}$ is covered by the net. This is the underlying idea of the algorithm.

Algorithm of Karp and Miller
Every vertex of the covering tree is labelled by an ω-marking and every arc is labelled by a transition. $AC(R, m_0)$ is initialised with a root r labelled by m_0. One manages in $\mathcal{TOHANDLE}$ (initialised with r) a subset of leaves of the current tree that must be examined.

While $\mathcal{TOHANDLE}$ is not empty **do**
  Extract q labelled by m from $\mathcal{TOHANDLE}$;
  For every transition t fireable from m **do**
    Create q_t son of q;
    Label the arc linking q to q_t with t;
    $m_t := m + C(t)$;
    $\Omega(m_t) := m_t$;
    For every place $p \in P$ **do**
      If q admits an ancestor (including itself) a labelled by m_a,
      with $m_a < m_t$ and $m_a(p) < m_t(p)$ **then**
        $\Omega(m_t)(p) = \omega$
      Endif
    Endfor
    Label q_t with $\Omega(m_t)$;
    If q does not admit an ancestor (including itself)
    with the same label as $q(t)$ **then**
      Insert q in $\mathcal{TOHANDLE}$
    Endif
  Endfor
Endwhile

First we show that this construction always terminates. Assume the contrary. Then the covering tree is infinite. Using lemma 2, this tree has an infinite branch. Lemma 3 holds for $(\mathbb{N}_\omega)^P$. So one can extract from this branch an increasing subsequence and even a strictly increasing one since otherwise the branch would be finite (see the insertion condition in $\mathcal{TOHANDLE}$). Applying arithmetic of $\mathbb{N}_\omega$, the ω-components cannot disappear on a branch. Assume that two consecutive vertices of the subsequence have the same set of ω-components. This means that in the second vertex, no ω-component has been created. But then, $\Omega(m(t)) = m(t)$ which is impossible since then $m(t)$ is strictly greater than the marking of the first vertex and there should have been a new ω-component. Otherwise stated, the number of ω-components is

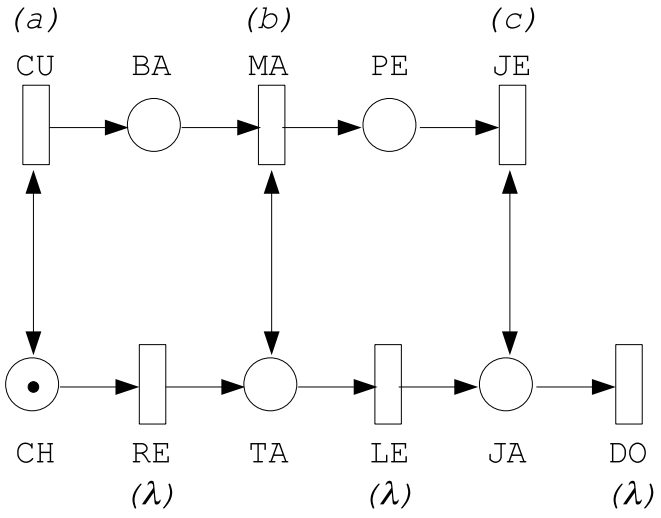

Figure 2: A day in the life of a banana planter

incremented at any marking of the subsequence. This leads to a contradiction since this number is bounded by $|P|$.

We illustrate the construction of the covering tree of the net of figure 2. This net models (in a very unrealistic way) a day in the life of a banana planter. Initially on his field (supposed to have an infinite capacity), the planter picks bananas, then he goes to his house with his bananas and he sits down to eat some of them. Once his meal is finished, he stands up and throws some banana skins in his garden before going to bed. We have indicated in parentheses a possible labelling of transitions. The language of the labelled net will be discussed at the end of the chapter.

The covering tree associated with this modelling is represented in figure 3. The vertices with an asterisk are those whose ω-marking is already present on the branch that leads to the vertex. Since there are leaves without asterisks, here one concludes that the net is not pseudo-live (be careful, this is only a sufficient condition). The mechanism of creation of an ω-component is triggered at the root. When one fires transition CU, one reaches a marking greater on place BA and identical on the other places which leads to the creation of an ω-component for BA. The ω-marquage $\overrightarrow{TA} + \omega.\overrightarrow{BA} + \omega.\overrightarrow{PE}$ illustrates an important point of the construction. One can reach a marking with an arbitrary large number of tokens in places BA and PE **but one cannot make them**

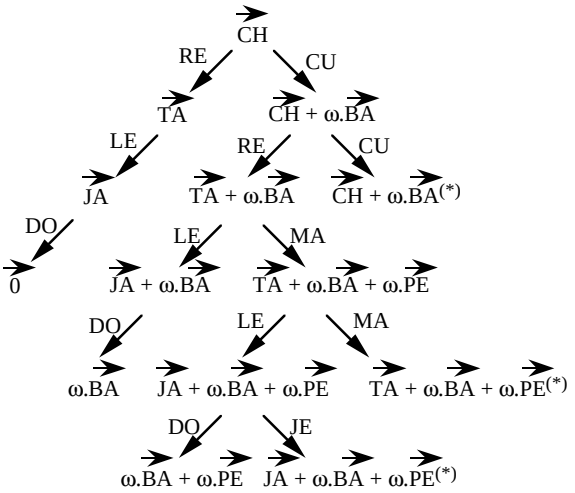

Figure 3: The covering tree of the *planter* net

simultaneously increase. One must first increase the marking BA, and then decreasing it while incrementing the marking of PE. This explains why the proof of decidability of covering is not so simple.

Proof
(of proposition 16)

Let m_c be a marking covered by the net. There exists a firing sequence $m_0 \xrightarrow{\sigma} m$ with $m \geq m_c$. Let us prove by recurrence that every marking of the sequence is covered by an ω-marking of $AC(R, m_0)$. This is the case for m_0. Assume that m_i is covered by $m_{\omega,i}$ and $m_i \xrightarrow{t} m_{i+1}$. Pick the vertex corresponding to the first occurrence of $m_{\omega,i}$. From this vertex, one builds a son corresponding to the firing of t labelled by $m_{\omega,i+1} \geq m_{\omega,i} + C(t) \geq m_i + C(t) = m_{i+1}$. This vertex remains in the tree except if there is a vertex with marking $m_{\omega,i+1}$ in the branch leading to this new vertex. Whatever the case, the recurrence is established.

Let q be a vertex of $AC(R, m_0)$ labelled by m_ω, a ω-marking. We prove that the markings smaller or equal than m_ω are covered by the net. More precisely, we show that for every $n \in \mathbb{N}$, there exists a reachable marking m_n equal to m_ω on its finite components and greater or equal than n on the other components.

We reason by recurrence on the number of ω-components of m_ω. If m_ω is an ordinary marking then the sequence of transitions that labels the branch

from r to q is a firing sequence and m_ω is reachable.

Suppose that the property holds for every vertex whose marking has at most d ω-components, and that q is the first vertex on the branch whose marking has $d+1$ ω-components.

- note $oldq$ the vertex preceeding q on the branch, whose ω-marking is $oldm$
- for $n \in \mathbb{N}$, note $\sigma_{0,n}$ the firing sequence that leads from m_0 to a marking m_n equal to $oldm$ on its finite components and greater or equal than n on the other ones (such a sequence exists by recurrence hypothesis)
- note t the transition which labels the arc from $oldq$ to q
- note $p_1, \ldots, p_k$ the places which are the new ω-components of m_ω
- note $r_1, \ldots, r_k$ the vertices, ancestors of q which justify the occurrence of these ω-components. Their associated ω-markings are noted $m_1, \ldots, m_k$
- note σ_i the sequence of transitions that labels the branch from r_i to q
- note $\Delta \in \mathbb{N}$ an integer such that the marking consisting of Δ tokens in every place enables to fire the sequences σ_i and transition t

Let us prove that the sequence $\sigma = \sigma_{0,(k.n+1).\Delta+n}.t.(\sigma_1)^n.\ldots.(\sigma_k)^n$ fulfils the searched property. We examine three kinds of places:

1. **w.r.t. to the finite components** m_ω, on the one hand, the sequence $\sigma_{0,(k.n+1).\Delta+n}.t$ leads to marking m_ω equal to these places to markings m_i. On the other hand, the sequences σ_i are repetitive stationnary on these places and fireable from m_i.
2. **w.r.t. the ω-components of** $oldm$, the sequence $\sigma_{0,(k.n+1).\Delta+n}$ leads to a marking where the marking of these places are greater or equal than $(k.n+1).\Delta+n$. By definition of Δ, one can fire the sequence $t.(\sigma_1)^n.\ldots.(\sigma_k)^n$ reaching a marking of these places greater or equal than n
3. **w.r.t. the new ω-components of** m_ω, the sequence $\sigma_{0,k.n.\Delta+n+1}.t$ leads to a marking greater of equal than any of m_i. For these places, every sequence σ_i is repetitive increasing (incrementing the marking of p_i). Hence on can fire $(\sigma_1)^n.\ldots.(\sigma_k)^n$ and obtain a marking of these places greater or equal than n.

If q is not the first vertex of this branch whose marking has $d+1$ ω-components, let q' with associated ω-marking m'_ω be such a marking. Let σ the sequence

that labels the branch from q' to q. As previously done, we define Δ in order to enable the firing of σ and we define $\sigma_{0,n}$ for m'_ω. The reader can check that the sequence $\sigma_{0,n+\Delta}.\sigma$ is appropriate. $\diamondsuit$

This tree is used in the proof of reachability (see later on). The main drawback of this construction is that the size of the tree is not primitive recursive. Indeed, in the case of bounded nets, it includes at least as vertices as reachable markings.

The covering graph

Several properties can be decided by examination of the covering tree or the *covering graph* obtained by identifying the vertices labelled with the same markings [VAL 85]. The operation consisting in "quotienting" the graphs is a useful construction that we describe in the framework of transition systems. We then apply this construction to obtain the covering graph from the covering tree.

Definition 17 (Transition systems) *A transition system $\mathcal{G}$ is defined by a tuple $< S, s_0, \rightarrow, L >$ where:*
- S denotes the set of states of $\mathcal{G}$,
- s_0, the initial state of S,
- L, the set of transition labels and
- $\rightarrow$ the transition relation ($\rightarrow \subset S \times L \times S$).

The reachability graph and the covering tree of a Petri net are transition systems whose labels are the transitions of the Petri net.

Definition 18 (Quotient of a transition system) *Let $\mathcal{G} =< S, s_0, \rightarrow, L >$ be a transition system and $\equiv$ be an equivalence relation included in $S \times S$, one denotes $\mathcal{G}_\equiv =< S_\equiv, s_{0\ \equiv}, \rightarrow_\equiv, L >$, the quotient of $\mathcal{G}$ by $\equiv$, defined as follows:*
- $S_\equiv$ represents the set of equivalence classes of $\equiv$,
- $s_{0\ \equiv}$ the equivalence class of s_0,
- $\rightarrow_\equiv$ the smallest set of $S_\equiv \times L \times S_\equiv$ fulfilling:
$[(s, l, s') \in \rightarrow] \Rightarrow [(s_\equiv, l, s'_\equiv) \in \rightarrow_\equiv]$

The above construction handles all labels in the same way. In the other chapters of the book, a more general construction is designed which takes into account a partition of labels between observable labels and unobservable ones.

Let q and q' be two vertices of the covering tree $AC(R, m_0)$ whose ω-markings are respectively m_ω and m'_ω. One defines the equivalence relation between vertices:

$$q \equiv q' \text{ iff } m_\omega = m'_\omega$$

Definition 19 *(Covering graph) The covering graph of a net, denoted* $GC(R, m_0)$*, is obtained by quotienting the covering tree* $AC(R, m_0)$ *by the equivalence relation* $\equiv$*.*

The following result is immediate.

Proposition 20 *If* (R, m_0) *is bounded then its covering graph and its reachability graph are identical: i.e.,* $GC(R, m_0) = G(R, m_0)$

The covering graph of the planter net (figure 2) is represented in figure 4.

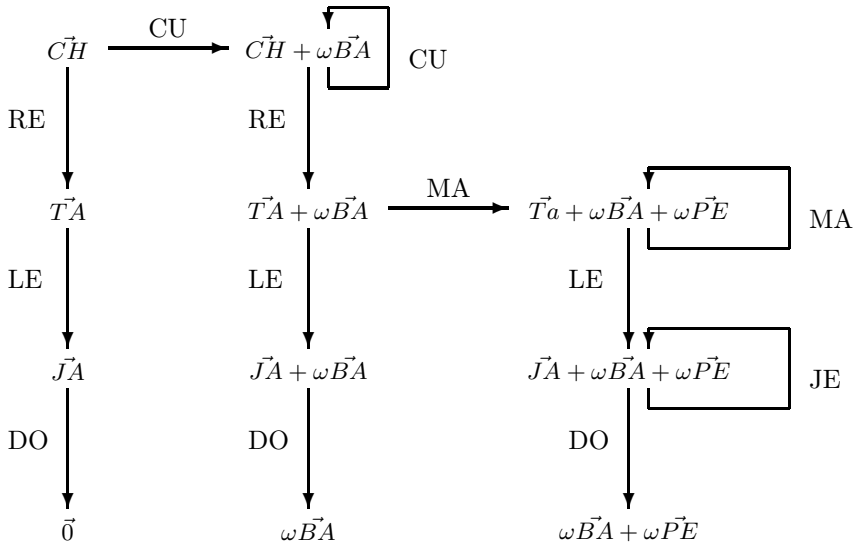

Figure 4: Covering graph of the planter net

In case of an unbounded net, the covering graph represents a superset of the real behaviour of the net. Without considering final markings, let us denote $LC(R, m_0)$ the language associated with the covering graph, then one has: $L(R, m_0) \subseteq LC(R, m_0)$.

This inclusion is often strict. In the example, the sequence $CU.RE.MA.MA$ of the covering graph is not a firing sequence: the planter cannot eat more bananas than the ones he has picked.

4.2. *Shortest sequences*

The principle of this method relies on the following result: if a marking can be covered by the net, then it can be covered using a firing sequence whose length is bounded by a computable function of the size of the net and of the marking to be covered [RAC 78]. Observe that the bound **does not depend on the initial marking**.

The algorithm consists in computing the bound and proceeding to a non deterministic search among the paths with length bounded by this bound.

We focus on the computation of the bound. Let us denote k the number of places, $P = \{p_1, \ldots, p_k\}$ and n the size of the covering problem (i.e. of its representation). Observe that $k \leq n$ and that every valuation of an arc and the marking m_c of every place are smaller or equal then 2^n (binary representation of an integer).

We reason on pseudo-firing sequences, i.e. such that the fireability condition is only partially fulfilled.

Definition 21 *Let $s = m_1.t_1.m_2.t_2 \ldots t_{y-1}.m_y$ be an alternated sequence of items of $\mathbb{Z}^k$ called pseudo-markings and transitions, let i be a place index between 0 and k:*

- *s is a i-firing sequence if*

$$\forall x \leq y-1,\ m_{x+1} = m_x + C(t_x) \text{ et } \forall j \leq i,\ m_x(p_j) \geq Pre(p_j, t_x) \text{ et } m_1(p_j) \geq 0$$

- *s is a i-firing sequence r-bounded if **moreover**:*

$$\forall x \leq y,\ \forall j \leq i,\ m_x(p_j) < r$$

- *s is a covering i-firing sequence if **moreover**:*

$$\forall j \leq i,\ m_y(p_j) \geq m_c(p_j)$$

Otherwise stated, to produce an i-firing sequence the test of fireability is only performed for the i first places but one consumes and produces tokens as usual which leads to negative values. Similarly, an i-firing sequence is r-bounded if every pseudo-marking of the sequence is bounded by r on the i first places. At last a covering i-firing sequence only covers the i first places. We say that (R, m) i-covers m_c if there exists a covering i-firing sequence whose initial marking is m.

Without taking into account m_0, let us call $lg(i, m)$ the length of the shortest covering i-firing sequence starting from a pseudo-marking m **counted as the**

number of pseudo-markings. This quantity is only defined for m such that (R, m) i-covers m_c. Let us note $f(i)$ the maximum of $lg(i, m)$ for every such m. A priori, $f(i)$ could be infinite. We remark that $f(k)$ is the bound we are looking for since in this case the sequence are real firing and covering sequences.

Since every (R, m) 0-covers m_c by the pseudo-firing sequence reduced to m, $f(0) = 1$. Our goal is to upper bound $f(i+1)$ by an expression involving $f(i)$.

Proposition 22

$$\forall\, 0 \leq i < k,\ f(i+1) \leq (2^n.f(i))^{i+1} + f(i)$$

Proof
Let m be any pseudo-marking such that (R, m) $(i+1)$-covers m_c and let s be a shortest covering $(i+1)$-firing sequence.

Case n ° 1 s is a $(i+1)$-firing sequence which is $2^n.f(i)$-bounded. For every shortest $(i+1)$-sequence, the pseudo-markings restricted to the $i+1$ first places must be different otherwise the sequence could be shortened by deleting a subsequence. The number of different restricted markings is equal to $(2^n.f(i))^{i+1}$ which yields the bound.

Case n ° 2 s is not a $(i+1)$-firing sequence which is $2^n.f(i)$-bounded. So $s = s_1.t.s_2$ with s_1 which is $2^n.f(i)$-bounded and s_2 which begins by a pseudo-marking m_2 with the marking of one of the $(i+1)$ first components which is at least equal to $2^n.f(i)$. W.l.o.g., one supposes that it is p_{i+1}. Using the proof of the case n ° 1, the length of s_1 is $\leq (2^n.f(i))^{i+1}$.

(R, m_2) $i+1$-covers m_c hence it i-covers m_c. This means that there exists an i-covering sequence s_2' with length at most $f(i)$ which starts from m_2. This sequence has at most $f(i) - 1$ transitions. The marking of p_{i+1} cannot decrease more than $2^n.(f(i) - 1)$ and so the final marking of p_{i+1} is $\geq 2^n \geq m_c(p_{i+1})$. Consequently s_2' is a covering $(i+1)$-firing sequence and the length of $s_1.t.s_2'$ fulfils the condition.

♢

By a straightforward recurrence, one obtains that $f(k) \leq 2^{(3n)^k} \leq 2^{2^{c.n.log(n)}}$ for a constant c independent of all parameters of the problem.

In order to non deterministically search a path, it is enough to only keep the last marking of the current path and a counter of its length. The marking cannot exceed $2^{2^{c.n.log(n)}}.2^n$ tokens in a place. This algorithm uses a memory size of $2^{d.n.log(n)}$ for another constant d.

By Savitch procedure [AHO 74], every algorithm can be determinised with a quadratic expense of the memory size. One obtains an algorithm in $\mathcal{EXP}space$.

The unboundedness of a net can be similarly solved (using a more elaborate proof) with the characterisation of proposition 4. These two problems are $\mathcal{EXP}space$-hard [LIP 76], so we obtain:

Theorem 23 (Complexity of covering and boundedness)
The covering and boundedness problems for a Petri net are $\mathcal{EXP}space$-complete.

In [YEN 92], this method is extended to a formula language for sequences, for which as soon as a formula is satisfiable, it is satisfiable by a sequence whose length is bounded by a computable value.

4.3. *Backward analysis*

The principle of backward analysis consists in building a finite representation of the set of initial markings that satisfy the covering property. It has first been designed in [ARN 76] and then rediscovered by [ABD 96] and at last generalised in [FIN 98].

Definition 24 *Let R be a Petri net and m_c be a marking,*

- *$Couv(R, m_c)$ is the set of markings m such that (R, m) covers m_c*
- *$\forall n \in \mathbb{N}, Couv_n(R, m_c)$ is the set of markings m such that (R, m) covers m_c by a sequence whose length is at most n.*

Before describing the algorithm, we gather some elementary facts.

- $Couv(R, m_c) = \cup_{n \in \mathbb{N}} Couv_n(R, m_c)$
- $Couv_n(R, m_c) \subset Couv_{n+1}(R, m_c)$
- $Couv_n(R, m_c)$ is upper closed: $m \in Couv_n(R, m_c)$ and $m' \geq m \Rightarrow m' \in Couv_n(R, m_c)$

Given a set of markings E, let us note $E^{\uparrow}$ the set of markings greater or equal than a marking of E. The upper closed sets are characterised by the property $E = E^{\uparrow}$.

We want to obtain $Couv(R, m_c)$ by iteratively computing $Couv_n(R, m_c)$. This requires that this sequence of sets stabilises, i.e. that after some index, all sets are equal and thus equal to $Couv(R, m_c)$.

Lemma 25 *Let $\{E_n\}_{n\in\mathbb{N}}$ be a sequence of upper closed increasing sets of markings; then this sequence stabilises.*

Proof
Assume the contrary, as soon as a set is not equal to the previous one, this means that there is a marking of this set which is not greater or equal than any of the marking of the previous set. We select such a marking. Iterating the process, one obtains an infinite sequence such that every marking is not greater or equal than any of the previous markings of the sequence. But this contradicts lemma 3. ♢

To obtain a finite representation of an upper closed set, it is sufficient to pick the minimal items of this set. Indeed, since they are all not comparable, there cannot be an infinite number of minimal items since again it would contradict lemma 3. Let us call $Min(E)$ the set of minimal items of E. Since E is upper closed, $E = Min(E)^{\uparrow}$ which shows that an upper closed set is wholly determined by its minimal items.

$Couv_0(R, m_c)$ represents the set of initial markings that cover m_c by a sequence with null length. Otherwise stated, they are the markings greater or equal than m_c. Hence $Min(Couv_0(R, m_c)) = \{m_c\}$.

We now show how to compute $Min(Couv_{n+1}(R, m_c))$ starting from $Min(Couv_n(R, m_c))$.

Let $m' \in Couv_{n+1}(R, m_c)$. By definition, m' covers m_c by a sequence of length at most n or equal to $n+1$. In the first case, m' is greater or equal than one of the markings of $Min(Couv_n(R, m_c))$. In the second case, it enables to fire a transition that leads to a marking greater or equal than a marking of $Min(Couv_n(R, m_c))$.

Let m be a marking of $Min(Couv_n(R, m_c))$. A marking obtained by firing of t greater or equal than m must necessarily be greater or equal than $Sup(Post(t), m)$ and the marking preceding the firing is greater or equal than $Sup(Post(t), m) - C(t)$. Summarising:

$$Min(Couv_{n+1}(R, m_c)) =$$
$$Min(Min(Couv_n(R, m_c)) \cup_{t\in T} \cup_{m\in Min(Couv_n(R,m_c))} Sup(Post(t), m) - C(t))$$

Every step multiplies by at most $|T| + 1$ the number of minimal items. The previous method gives a bound on the number of stages before stabilisation. So one deduces that the method is primitive recursive.

A priori, the shortest sequence method seems more efficient. However the interest of backward analysis will be illustrated during the presentation of extensions of Petri nets.

5. The reachability problem

The reachability problem consists in deciding whether, given a net R and two markings m_i and m_f, there exists a firing sequence: $m_i \xrightarrow{\sigma} m_f$. The decision algorithm has been independently established by E. Mayr [MAY 84] and S.R. Kosaraju [KOS 82]. Here we will not fully describe this proof whose some features are very technical and to which a book is devoted [REU 89]. We present the scheme of the proof and we only develop its main feature, i.e. the sufficient condition for reachability.

First, the reachability problem is solved for a more general model than the one of Petri nets, called chain of vector addition systems ($CVAS$). The main motivation of this generalisation is intimately linked to the proof. This proof can be described as follows:

- A *size* is defined for the reachability problem. This size is an item of a well founded set (i.e. such that **there does not exist an infinite sequence of strictly decreasing sizes**).
- One establishes a sufficient condition for reachability with the help of **the covering graph**.
- If the condition is not fulfilled, one builds a finite (possibly empty) set of problems **with smaller sizes** such that there exists a solution for the initial problem iff there exists a solution for any of the reduced problems.
- Taking the initial problem as root, one builds a tree of problems defining for the sons of a problem, its reduced problems. If the tree was infinite, there would be an infinite branch which is impossible due to the first point.
- The initial problem has a solution iff some leaf has a solution (checked by the sufficient condition).

This procedure deserves some comments. Since it is based on the covering graph, it is not primitive recursive. On the other hand, one only knows that the problem of reachability is $EXPspace$-hard [LIP 76]. Otherwise, an open issue is to determine whether this problem is primitive recursive. The reduction of the problem in smaller problems of the same kind explains the choice of the model $CVAS$. As we will see later on, in Petri nets when the sufficient condition is not fulfilled, one reduces the problem to different kinds of problems.

The decision problem of several properties can be reduced more or less directly to the reachability problem. For instance, the liveness problem [HAC 75],

the pseudo-liveness problem and, given a marking, the home state problem are decidable [FRU 86, FRU 89].

We begin the presentation of the sufficient condition by a decision procedure for a necessary condition that is part of the sufficient condition.

5.1. *A necessary condition for reachability*

If there exists a sequence σ such that $m_i \xrightarrow{\sigma} m_f$ then, due to the equation of state change, one has: $C.\overrightarrow{\sigma} = m_f - m_i$. Let us show how to decide the existence of a solution $x \in \mathbb{N}^T$ of equation $C.x = b$ with $b \in \mathbb{Z}^P$. In case where $|T| = 1$, the equation can be trivially solved. So we assume that $|T| > 1$.

First we apply the computation of T-flows. If no T-flow is found, then the column vectors $\{C(t)\}_{t \in T}$ form a linearly independent family and there exists at most one solution in $\mathbb{Q}^T$ to this problem. The existence and the computation of this solution is performed by the main variation of Gauss elimination. If a solution is obtained, one checks that it belongs to $\mathbb{N}^T$.

In the other case, let us call a the T-flow computed ($C.a = \overrightarrow{0}$), and let us note $T' = \{t \mid a(t) > 0\}$. W.l.o.g. we assume that T' is not empty. Assume that there exists a solution x to the equation such that $\forall\, t \in T',\ x(t) > a(t)$ then $x - a$ is also a solution. Iterating this process, one obtains y a solution fulfilling $\exists\, t \in T'$ t.q. $y(t) \leq a(t)$. One can substitute to the original equation a family of equations, one per pair $t \in T'$ and $0 \leq i \leq a(t)$ where one substitutes variable $x(t)$ by i. The existence of a solution for some equation is equivalent to the existence of a solution for the initial equation. In every new equation a variable has disappeared. By iteration, this leads (in the worst case), to equations with a single variable whose resolution is straightforward.

5.2. *A sufficient condition for reachability*

We introduce the reverse net $\widetilde{R}$ which fires transition of R backwards. Places and transitions of this net are the ones of R and the incidence matrices are defined by: $\forall\, t \in T,\ \widetilde{Pre}(t) = Post(t)$ and $\widetilde{Post}(t) = Pre(t)$

An elementary check gives:

$\forall\, \sigma \in T^*\ m \xrightarrow{\sigma}_R m' \Leftrightarrow m' \xrightarrow{\widetilde{\sigma}}_{\widetilde{R}} m$, $\widetilde{C} = -C$ and $\overrightarrow{\widetilde{\sigma}} = \overrightarrow{\sigma}$

Our goal is to transform the previous condition into a sufficient condition. Reusing the vocabulary of section 4.2, we call pseudo-marking an item of $\mathbb{Z}^P$. Given two pseudo-markings m, m' and a sequence of transitions σ, we call a

pseudo-firing sequence (denoted $m(\sigma\rangle m'$), a sequence which fulfils the state change equation $m' = m + C.\overrightarrow{\sigma}$.

We first establish a preliminary result about conditions which ensure that a pseudo-firing sequence is a firing sequence.

Lemma 26 *Let R be a Petri net and $m_0(\sigma\rangle m_1(\sigma\rangle m_2 \ldots m_{k-1}(\sigma\rangle m_k$ be a pseudo-firing sequence then:*

$$m_0(\sigma\rangle m_1 \text{ and } m_{k-1}(\sigma\rangle m_k \text{ are firing sequences}$$
$$\Leftrightarrow \text{ The whole sequence is a firing sequence}$$

Proof
We prove the implication since the reverse implication is trivial. We partition P in two subsets $P' = \{p \in P \mid \overrightarrow{p}^t.C.\overrightarrow{\sigma} \geq 0\}$ and $P'' = \{p \in P \mid \overrightarrow{p}^t.C.\overrightarrow{\sigma} < 0\}$. The sequence σ restricted to P' is repetitive for net R, thus since $m_0 \xrightarrow{\sigma}_R m_1$ one has $m_0 \xrightarrow{\sigma^k}_{R,P'} m_k$.

Applying the elementary results about R and $\widetilde{R}$, the sequence $\widetilde{\sigma}$ restricted to P'' is repetitive increasing for net $\widetilde{R}$. Thus since $m_k \xrightarrow{\widetilde{\sigma}}_{\widetilde{R}} m_{k-1}$ one has $m_k \xrightarrow{\widetilde{\sigma}^k}_{\widetilde{R},P''} m_0$ which is equivalent to $m_0 \xrightarrow{\sigma^k}_{R,P''} m_k$.

Since $P = P' \cup P''$, we obtain: $m_0 \xrightarrow{\sigma^k}_R m_k$. ♢

To obtain a sufficient condition, one would show that the fireability conditions are somewhat irrelevant. Roughly speaking, the main idea consists to increase the initial marking by a firing sequence (σ_1^k in the proof) such that the pseudo-firing sequence (σ_2) becomes a firing sequence, and then decrease the marking by another firing sequence (σ_4^k). However before this last firing sequence, one inserts an intermediate sequence (σ_3^k) in order to cancel the cumulated effects of the first and last sequence.

Proposition 27 (Sufficient condition) *Let R be a Petri net, m_i and m_f two markings, if:*

1. *R is consistent*
2. *$\exists\, v \in \mathbb{N}^T$ t.q. $C.v = m_f - m_i$*
3. *$\sum_{p \in P} \omega.\overrightarrow{p}$ belongs to the covering trees $AC(R, m_i)$ and $AC(\widetilde{R}, m_f)$*

Then m_f is reachable from m_i in the net R.

Proof
We build step by step the firing sequence.

First condition 3 implies the existence of an increasing repetitive sequence σ_1 (for every place) from m_i dans R and an increasing repetitive sequence $\widetilde{\sigma_4}$ (for every place) from m_f in $\widetilde{R}$.

Since R is consistent there exists a positive vector w with support T such that $C.w = 0$. Since its support is T, there exists an integer n enough great such that $w' = n.w - \overrightarrow{\sigma_1} - \overrightarrow{\sigma_4} \geq \overrightarrow{0}$. Let us denote σ_3 an arbitrary sequence which fulfils $\overrightarrow{\sigma_3} = w'$.

Finally, let us denote σ_2 an arbitrary sequence fulfilling $\overrightarrow{\sigma_2} = v$. Observe that σ_2 is a pseudo-firing sequence from m_i to m_f. Let $k \geq 2$ an integer such that the marking of all places with k tokens makes the following sequences fireable:

- the sequence $\sigma_2.\sigma_3$ in R
- and the sequence $\widetilde{\sigma_3}$ in $\widetilde{R}$

We claim that $m_i \xrightarrow{\sigma_1^k} m_1 \xrightarrow{\sigma_2} m_2 \xrightarrow{\sigma_3} m_3 \xrightarrow{\sigma_3^{k-2}} m_4 \xrightarrow{\sigma_3} m_5 \xrightarrow{\sigma_4^k} m_f$ is a firing sequence (we introduce intermediate markings to ease the proof).

Let us compute the incidence of this sequence:

$$C.(k.\overrightarrow{\sigma_1} + \overrightarrow{\sigma_2} + k.\overrightarrow{\sigma_3} + k.\overrightarrow{\sigma_4}) = C.(v + k.(w' + \overrightarrow{\sigma_1} + \overrightarrow{\sigma_4}))$$

$$= C.(v + k.n.w) = C.v = m_f - m_i$$

So this sequence is a pseudo-firing sequence. Let us show that the fireability conditions are met (we implicitely use the relations between R et $\widetilde{R}$):

- by definition of σ_1 and σ_4, we have $m_i \xrightarrow{\sigma_1^k} m_1$ and $m_5 \xrightarrow{\sigma_4^k} m_f$,
- by the choice of k, we have $m_1 \xrightarrow{\sigma_2} m_2 \xrightarrow{\sigma_3} m_3$ and $m_4 \xrightarrow{\sigma_3} m_5$,
- which implies due to the previous lemma $m_2 \xrightarrow{\sigma_3} m_3 \xrightarrow{\sigma_3^{k-2}} m_4 \xrightarrow{\sigma_3} m_5$.

$\diamondsuit$

Let us examine how to carry on with the decision procedure if one point of the sufficient condition is not fulfilled. If point 2 is not fulfilled (i.e the necessairy condition) then m_f is not reachable. If point 3 is not fulfilled then it means that during the possible firing sequence, the marking of a place remain bounded by its greatest finite value occurring on the covering trees (in fact the

minimum of its greatest finite values on each tree). Hence one substitutes to the initial problem, $|P|$ reachability problems with a bounded place by a known value.

For the case of inconsistency, we introduce particular subsets of vectors with positive integer coefficients.

Definition 28 (Semi-linear sets) *A* linear *set of positive integer vectors E is defined by a vector u and a family of vectors $V = \{v_1, \ldots, v_m\}$:*

$$E = \{w \mid \exists\, \lambda_1, \ldots, \lambda_m \ \in \mathbb{N}, t.q.\ w = u + \textstyle\sum_{i=1}^{m} \lambda_i.v_i\}$$

A semi-linear set of positive integer vectors E is a finite union of linear sets.

Semi-linear sets are finite representations of infinite sets which have numerous interesting properties. One can compute the union and the intersection of two semi-linear sets and the complementary of a semi-linear set; all these sets are semi-linear. Furthemore one can decide whether a vector belongs to a semi-linear set. If the reachability set was an effective semi-linear set of $\mathbb{N}^P$ the reachability problem would be solved. Unfortunately, the reachability sets of some nets are not semi-linear [HOP 79] and one can decide whether it is the case [HAU 90].

However the set E of solutions of $C.x = m_f - m_i$ is a semi-linear set whose representation is computable (see for instance [REU 89]) and more precisely equal to $\{u + \sum \lambda_k.v_k \mid u$ is a minimal solution of $C.x = m_f - m_i$, $\lambda_k \in \mathbb{N}$ and $\{v_k\}$ is the set of minimal solutions of $C.x = \overrightarrow{0}\}$; the number of these minimal solutions being finite due to lemma 3. Consequently, the net is not consistent iff there exists t such that $v(t) = 0$ for every solution v of $C.x = \overrightarrow{0}$. Otherwise stated, every reachability sequence would have a number of occurrences of t equal to some value $u(t)$ of a minimal solution of $C.x = m_f - m_i$. We replace the reachability problem by a set of problems for which the reachability sequence has a fixed number of occurrences of some transition.

Intuitively in both transformations, the *infinite* character of the problem has been reduced since in one case the marking of a place is bounded and in the other case the number of occurrences of a transition is bounded. The formalisation of this argument yields the model of *CVAS*.

6. Extensions of Petri nets

6.1. *Nets with inhibitor arcs*

The expressive power of Petri nets is close to the one of a programming langage working on integers. However the nets miss the capability to test equality between the marking of a place and some constant. To this aim, nets with inhibitor arcs have been introduced. In this model, the incidence matrices are completed by an inhibition matrix which enforces an upper bound on the marking of some place in order to fire a transition.

Definition 29 (Petri nets with inhibitor arcs) *A Petri net with inhibitor arcs is defined by a tuple* $R = \langle P, T, Pre, Post, Inh \rangle$ *where:*

- P *is a finite set of places,* T *is a finite set of transitions.*
- Pre *and* $Post$ *are the backward and forward incidence matrices defined in* $\mathbb{N}^{P \times T}$.
- Inh *is the inhibition matrix defined in* $(\mathbb{N}_\omega \setminus 0)^{P \times T}$

Definition 30 (Firing rule with inhibitor arcs)
Let m *be a marking of a Petri net with inhibitor arcs and* t *be a transition:*

- t *is fireable from* m *iff*
$$m \geq Pre(t) \text{ and } m < Inh(t) \text{ (component per component)}$$
- *The firing of* t *from* m *leads to marking* m' *defined by:*
$$m' = m + C(t)$$

Only the fireability condition is modified. An inhibitor arc is represented by a line ended with a small circle. The valuations different from 1 label the arcs and valuation ω is not represented (since it does not restrict the behaviour of the net).

Figure 5 explains the main difference between nets with inhibitor arcs and ordinary nets. If one adds to the initial marking a tokens in p_a and b tokens in p_b, then the single maximal sequence $(t_1.t_2^b.t_3.t_4^b.t_5)^a$ leads to a marking where p_c has $a.b$ tokens. If the inhibitor arcs are omitted, there are several maximal sequences, all leading to a number of tokens p_c less or equal than $a.b$ and one of them fulfilling the equality. Otherwise stated, a Petri net *weakly* computes (in the sense described above) any computable increasing arithmetical function while a net with inhibitor arcs *exactly* computes any comptable arithmetical

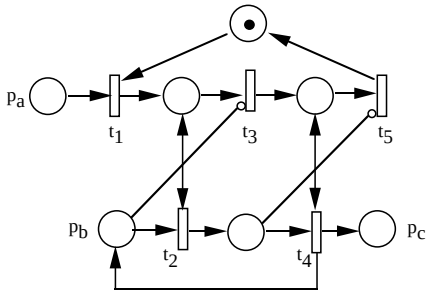

Figure 5: Product of two numbers using a net with inhibitor arcs

function The net of figure 2 also illustrates the difference of expressive power. In this net (**and in every net** modelling the actions of the planter), it is impossible to garantee that the planter eats all his bananas before going to the garden while adding a single inhibitor arc is enough to obtain this behaviour.

The extended expressive power leads to the undecidability of all interesting properties (reachability, liveness, boundedness, covering, termination, ...). Indeed, the stop of a program is an undecidable problem and it is not difficult to reduce this problem to the decision problem of one of these properties even with two inhibitor arcs.

However, it has been proved that the reachability remains decidable with a single inhibitor arc or with an "ordered" structure of inhibitor arcs w.r.t. the places ($Inh(p_i, t) \neq \omega$ and $j < i \Rightarrow Inh(p_j, t) \neq \omega$), (all these arcs are valuated by 1) [REI 95].

6.2. *Self-modifying nets*

Another interesting extension consists in making the number of tokens to consume or produce depend on the current marking. In the initial model of self-modifying nets, the valuation of an arc is a linear combination of place markings plus some constant [VAL 78]. In the generalised model called G-nets, the valuation is a polynomial with non negative coefficients on places [DUF 98b].

Notation

- $\mathbb{N}[P]$ denotes the set of polynomial with non negative coefficients whose variables are places.
- Let m be a marking and Q be such a polynomial, then $Q[m]$ denotes the value of the polynomial when the value of every variable p equals $m(p)$.

Definition 31 (G-net)
A G-net is defined by a tuple $R = \langle P, T, Pre, Post \rangle$ *where:*

- P *is a finite set of places,* T *is a finite set of transitions.*
- Pre *and* $Post$ *are backward and forward incidence matrices defined in* $\mathbb{N}[P]^{P \times T}$.

Definition 32 (Firing rule in G-nets) *Let* m *be a marking of a G-net and* t *be a transition:*

- t *is fireable from* m *iff :*
$$m \geq Pre(t)[m]$$
- *The firing of* t *from* m *leads to marking* m' *defined by:*
$$m' = m - Pre(t)[m] + Post(t)[m]$$

Without important restrictions, G-nets easily simulate the inhibitor arcs and consequently the main properties are undecidable. The easiest way to simulate an inhibitor arc from p to t is to define $Pre(p,t) = 2.p$ since $m(p) \geq 2.m(p) \Leftrightarrow m(p) = 0$.

This simulation gives hints about restrictions to apply in order that some properties remain decidable. We indicate three restrictions and recommend the thesis of C. Dufourd [DUF 98a] for a detailed study about the hierarchy of restrictions.

- In *G-Post-nets* the valuations of preconditions are integers. From the definition of the firing rule, the two assertions of monotonicity lemma 1 hold. Thus the construction of the covering tree is still possible (with some adaptation). Hence one can decide the covering, the boudedness of a net, the boundedness of a place and the termination.
- In *G-post-nets with reset* the valuation of an arc from p to t is either an integer, or the polynomial p. In this last case, firing t empties place p. In these nets, only the first assertion of lemma 1 hold:
$$\forall\, m_1 \leq m'_1\ m_1 \xrightarrow{\sigma} m_2 \Rightarrow\ m'_1 \xrightarrow{\sigma} m'_2 \text{ avec } m_2 \leq m'_2$$
One can still decide termination by a construction which detects the repetitive sequences. The covering is also decidable but then **with the backward analysis of section 4.3** which only relies on the first assertion of lemma 1. An important and difficult result is the undecidability of boundedness.
- The *G-post-nets with transfer* is a restriction of the previous model where the presence of an arc from p to t labelled by p implies the presence of an

arc from t to some place p' labelled by $p+Q$ where $Q \in \mathbb{N}^{[P]}$. Otherwise stated, when one empties the content of a place, it is moved in another place (with possibly additional tokens). Here, the construction of the covering tree correctly detect that the net is unbounded when the first ω occurs. It is amazing to observe that the place boundedness problem is undecidable. Indeed, one reduces the boundedness problem for a net with reset to the problem of boundedness of a subset of places by a straightforward transformation. One adds to the net with reset a place *sink* and for every transition t, an arc from t to *sink* labelled by $\sum_{p \in P'} p$ where P' is the set of places with a reset arc to t. It is immediate that the net with reset is bounded *iff* every place of $P \setminus \{sink\}$ is bounded in the new net.

We end the overview of these models by showing that reachability is undecidable in presence of:

- either output arcs labelled by the destination place called double arcs (for obvious reasons)
- or reset arcs.

Proposition 33 *The reachability problem in Petri nets with inhibitor arcs is reducible to:*

- *The reachability problem in G-nets with only ordinary arcs and double arcs,*
- *The reachability problem in G-nets with only ordinary and reset arcs.*

Proof
For every place p of the net with inhibitor arcs, we add a place p^+ with the same incidences as p. Let m be a marking of the first net, then m^+ in the second net is defined by $m^+(p) = m^+(p^+) = m(p)$. The inhibitor arcs of the first net are transformed as indicated in figure 6 either with a double arc, or with a reset arc. The reader can check that whatever the construction:

$$m' \text{ is reachable from } m \text{ in the first net}$$
$$\Leftrightarrow m'^+ \text{ is reachable from } m^+ \text{ in the second net}$$

Indication: p^+ contains the same number of tokens as p iff there have been no firing of transition with an inhibitor arc from p (in the initial net) while p was marked. ♢

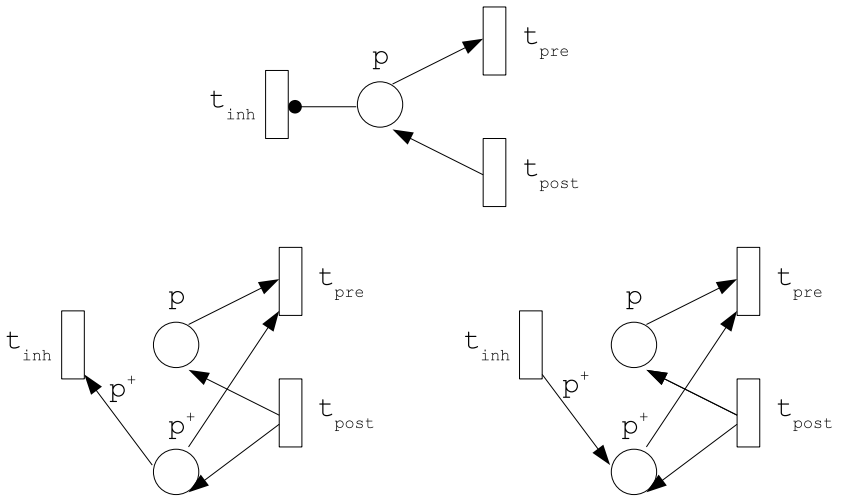

Figure 6: "Simulation" of an inhibitor arc

6.3. *Recursive nets*

A recursive net [HAD 99b, HAD 07] has the same structure as the one of a Petri net, except that in recursive nets, transitions are partitioned in two categories: abstract transitions and elementary ones. Furthermore, a *start* marking is associated with every abstract transition and a semi-linear set of final markings is defined. The semantic of such a net can be informally explained as follows. In a Petri net, a process *plays* with tokens, firing a transition and updating the current marking. In a recursive net, there is a dynamical tree of processes corresponding to fatherhood relation; every process playing its own token game. A step of a recursive net is then an execution step of any process. If the process fires an elementary transition, it udpdates its current marking using the ordinary firing rule. If the process fires an abstract transition, it consumes the tokens of preconditions of the transition and generates a new son which starts its token play with the start marking of the transition. If the process has reached a terminal marking, it can terminate killing all its descendants and producing in the marking of its father, the tokens of postconditions of the transition whose firing has triggered its creation. If it is the root process, one obtains an empty tree. We formalise this behaviour in the following definitions.

Definition 34 (Recursive net) *A recursive net is defined by a tuple* $R = \langle P, T, Pre, Post, \Omega, \Upsilon \rangle$ *where:*

- P *is a finite set of places,* T *is a finite set of transitions.*
- *A transition of* T *is either elementary or abstract. The subsets of elementary and abstract transitions are respectively denoted by* T_{el} *and* T_{ab}*.*
- Pre *and* $Post$ *are backward and forward incidence matrices defined in* $\mathbb{N}^{P \times T}$*.*
- Ω *is a function which associates with every abstract transition an ordinary marking (i.e. an item of* $\mathbb{N}^P$*) called the start marking of* t*.*
- Υ *is an effective semi-linear set of terminal markings*

An effective representation of a semi-linear set is a representation which can (by an algorithm) be transformed as the one of definition 28. For instance, a linear (in)equation over markings is an effective representation.

Definition 35 (Extended marking) *An* extended marking *tr of a recursive net R is a labelled tree $tr = \langle V, M, E, A \rangle$ where:*

- V *is the set of vertices,* M *is a function* $V \mapsto \mathbb{N}^P$*,*
- $E \subseteq V \times V$ *is the set of arcs and* A *is a function* $E \mapsto T_{ab}$*.*

A marked recursive net (R, tr_0) is a recursive equipped with an initial extended marking.

Let v be a vertex of an extended marking, $pred(v)$ denotes its father in the tree (defined if v is not the root) and $Succ(v)$ the set of direct and indirect successors (including v). An *elementary step* of a recursive net is, either a transition firing, or the deletion of a subtree (named *termination step* and denoted by τ).

Definition 36 *A transition t is* fireable *in a vertex v of an extended marking tr (denoted by $tr \xrightarrow{t,v}$) if $M(v) \geq Pre(t)$ and a termination step is* fireable *in v (denoted by $tr \xrightarrow{\tau,v}$) if $M(v) \in \Upsilon$*

Definition 37 *The* firing *of a fireable elementary step t in a vertex v of an extended marking tr leads to marking tr' defined w.r.t. the type of t.*

- $t \in T_{el}$

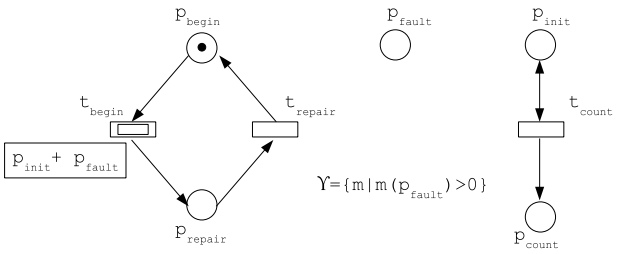

Figure 7: A fault-tolerant system

- $V' = V$, $E' = E$, $\forall e \in E, A'(e) = A(e)$, $\forall v' \in V \setminus \{v\}$, $M'(v') = M(v')$
- $M'(v) = M(v) - Pre(t) + Post(t)$

- $t \in T_{ab}$, *(v' is a new identifier thus not present V)*
  - $V' = V \cup \{v'\}$, $E' = E \cup \{(v, v')\}$, $\forall e \in E, A'(e) = A(e)$, $A'((v, v')) = t$
  - $\forall v'' \in V \setminus \{v\}, M'(v'') = M(v'')$, $M'(v) = M(v) - Pre(p)$
  - $M(v') = \Omega(t)$

- $t = \tau$
  - $V' = V \setminus Succ(v)$, $E' = E \cap (V' \times V')$, $\forall e \in E', A'(e) = A(e)$
  - $\forall v' \in V' \setminus \{pred(v)\}, M'(v') = M(v')$
  - $M'(pred(v)) = M(pred(v)) + Post(A(pred(v), v))$

  If v is the root of the tree then the firing of τ leads to the empty tree denoted $\perp$.

At first sight, it seems that associating the same net with every abstract transition is somewhat restrictive. En reality, it is easy to simulate a net with the activation of a net depending on the abstract transition. Using a single net alleviates the notations and eases the proofs. Most of usual conditions can be described by a semi-linear set. For instance, one can specify the set of dead markings, the fireability of a transition, mutual constraints on some place markings, ... We illustrate now the expressive power of this model using a simple modelling. Some other relevant examples are described in [HAD 00]. We represent an abstract transition by a rectangle with a double border equipped with its start marking inside a frame.

In order to study fault-tolerant systems, the engineer starts with a description of the functional system and then introduces the faulty behaviours and the mechanisms of repair. Here the functional system periodically records a measure from the environment (elementary transition t_{count}). The number of measures is stored in place p_{count}. The complete system is obtained by adding the left part of figure 7. The behaviour of this recursive net is the following one. Initially and immediately after the occurrence of a fault, the extended marking is reduced to a single vertex. A token in place p_{repair} indicates that the system is repairing while a token in place p_{init} means that the system is ready. When abstract transition t_{begin} is fired, the correct behaviour of the du system is executed by the new process. The termination of this process represents an occurrence of a fault. As place p_{fault} is always marked in this second vertex and due to the definition of Υ, the occurrence of a fault is always possible. Adding some places and updating Υ, we could model more complex fault patterns (e.g. faults triggered by software execution).

The state of the net is, either a tree with a single vertex, or a tree with a root and a leaf. However the number of reachable markings in this leaf is infinite. This means that the faulty state can be reached by an infinite number of states. This modelling is **impossible** with a Petri net since a state can only reached from at most $|T|$ transitions. The self-modifying nets also have this capability but not nets with inhibitor arcs.

Contrary to other extensions, the two main decidable properties of Petri nets are also decidable for recursive nets: reachability and boundedness [HAD 99a].

7. Languages of Petri nets

The introduction of "extended" Petri nets aims to increase the expressive power of nets while preserving decidability of some properties. The families of languages generated by nets are one of the means to determine this expressive power. Initially, formal languages have been studied in relation with grammars [HOP 69]. Let us briefly recall that a grammar includes non terminal symbols (with an initial symbol) and terminal ones (the characters of the alphabet). A grammar is composed of transformation rules $\{S_1 \dots S_m \rightarrow T_1 \dots T_n\}$. To compose a word of the language associated with a grammar, one starts with the initial symbol and one applies any transformation rule to a subword of the current word until the word has only terminal symbols.

Depending on the structure of grammars, one defines families of languages and one studies problems like:

- the membership of a word to the language,

- the emptyness of the language,
- the closure of a family of languages under operations like union, intersection and complementary.

Each problem has an interpretation w.r.t. the behaviour of systems modelled for instance by Petri nets. The membership problem is related to the test: whether expected behavioural sequences really occur. The emptyness problem is related (with an appropriate choice of final markings) to the existence of at least a faulty sequence. The closure of a family by operations offers the possibility to the designer to modularly build systems using specifications given by such operations. For instance, the intersection of languages very often corresponds to a synchronisation between subsystems.

Usually, one distinguishes four families of languages strictly nested. Regular languages are generated by grammars whose rule patterns are: $S \rightarrow \lambda$, $S \rightarrow a.T$, $S \rightarrow a$ with S,T non terminal and a terminal. Algebraic languages are generated by grammars whose rule patterns are: $S \rightarrow T_1 \ldots T_n$ with n possibly null. Context-sensitive languages are are generated by grammars whose rule patterns are: $S_{init} \rightarrow \lambda$, $S_1 \ldots S_m \rightarrow T_1 \ldots T_n$ with $n \geq m$ and where S_{init} is the initial symbol. At last, type 0-langages have no restriction on rule patterns.

Example 1 *A regular and an algebraic grammar*
The following grammar denotes the behaviour of a process iterating an action until it succeeds:
$S \rightarrow try.T$, $T \rightarrow fail.S$, $T \rightarrow success$
The associated language L is defined by:
$L = \{try.(fail.try)^n.success\}_{n \in \mathbb{N}}$ also noted in a compact way
$L = try.(fail.try)^*.success$

The following algebraic grammar generates the language L' of palindromes on alphabet $\Sigma = \{a, b\}$, $L' = \{\sigma \in \Sigma^* \mid \widetilde{\sigma} = \sigma\}$
$S \rightarrow \lambda$, $S \rightarrow a$, $S \rightarrow b$, $S \rightarrow a.S.a$, $S \rightarrow b.S.b$

One can decide the membership problem for the three first families and this problem is undecidable for type 0-languages. One can decide the emptyness problem for regular and algebraic languages but this problem is undecidable for context-sensitive languages. At last, regular languages are closed by the standard operations while for instance, the intersection of two algebraic languages is not necessarily an algebraic language [AUT 87].

The theory of languages of Petri nets consists in analysing the same kind of problems and in positioning the languages of Petri nets w.r.t. the standard families [PET 81]. For sake of readability, we recall here the definition of a Petri net language. Then we indicate the main results.

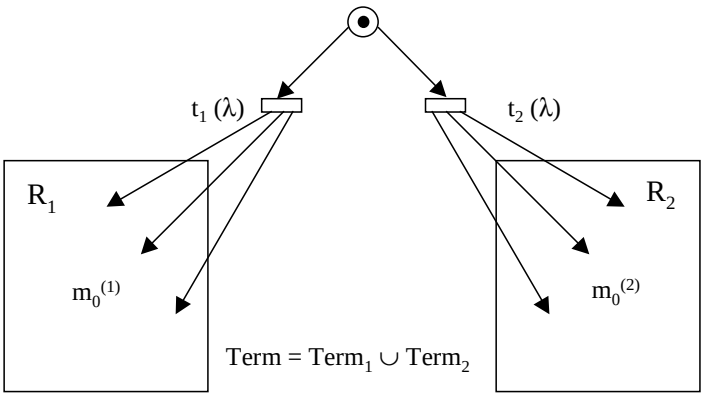

Figure 8: Construction of a net for the union of languages

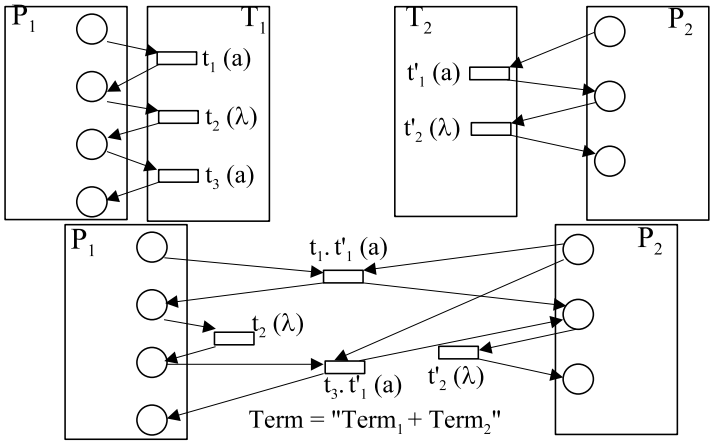

Figure 9: Construction of a net for the intersection of languages

Definition 38 (Language of a net) *Let (R, m_0) be a Petri net, Σ be an alphabet and l be a labelling mapping from T to $\Sigma \cup \lambda$ (the empty word). The labelling mapping extends to sequences by $l(\lambda) = \lambda$ and $l(\sigma.t) = l(\sigma).l(t)$. Let $Term$ be a finite set of terminal markings. The language of the net denoted $L(R, m_0, l, Term)$ is defined by:*

$$L(R, m_0, l, Term) = \{w \in \Sigma^* \mid \exists\, \sigma \in T^*, \exists\, m_f \in Term,\ m_0 \xrightarrow{\sigma} m_f \text{ and } w = l(\sigma)\}$$

Proposition 39 (Closure properties for Petri nets) *Languages of Petri*

nets are closed by union and intersection.

Proof
The construction of a net that accepts the union of Petri net languages is presented figure 8. One inserts the two nets (we assume there are disjoint) without initial marking. One adds to the net a place initially marked input of two new (initially fireable) transitions labelled by the empty word and whose outputs are the initial markings of the two nets. The set of terminal markings is the union of the two sets of terminal markings (or more precisely their mappings in the new vector space of place markings). This net non deterministically chooses to trigger one of the two nets that will produce a word of its language.

The construction of a net that accepts the intersection of Petri net languages is presented figure 9. One inserts the places of the two nets (we assume there are disjoint) with their initial marking. For every pair of transitions (one per net) labelled by the same character, one creates a transition with this label whose incidences are the sum of incidences of each transition. The transitions labelled by the empty word are added without any change. A terminal marking is the sum of a terminal marking of the first net and a terminal marking of the second net. So a word is simultaneously accepted in both nets, since every character is produced by the simultaneous firing of a pair of transitions. Transitions labelled by the empty word need to be fired independently in order to generated every possible subsequence that produces the empty word in any of the two nets. ♢

Proposition 40 (Analysis of Petri net languages) *The membership problem and the emptyness problem are decidable for Petri net languages.*

Proof
To check the (non) emptyness of the language, one decides whether one of the terminal marking is reachable. To check wether a word belongs to the language of a Petri net, one builds a second net that only accepts this word and then the net that accepts the intersection of net languages. Last one checks the emptyness of its language. ♢

Proposition 41 (Position of Petri net languages) *The family of Petri net languages contain the regular languages and is incomparable with the family of algebraic languages [JAN 79].*

Proof
To simulate regular grammar by a net, one associates with every non terminal

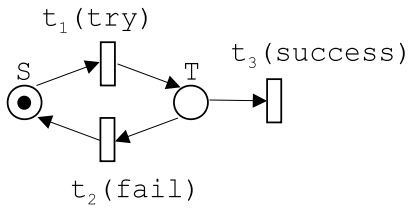

Figure 10: Simulation of a regular grammar

symbol a place and with every rule, a transition labelled by the terminal symbol of the rule, whose precondition is the symbol of the lefthand member of the rule et and postcondition is the symbol of the righthand member of the rule if it exists. The place of the initial symbol initially contains a token (and it is the only one) and the terminal marking is the null marking. A simulation of the grammar of example 1 is represented figure 10.

Observe that the language of net of figure 2 (the planter net) whose final marking is the null marking ($\{a^n.b^n.c^n \mid n \geq 0\}$) is not an algebraic language (due to Ogden lemma [AUT 87]). On the other hand, one proves that the language of palindromes is not a language of Petri nets (but the proof is rather technical). ♢

Proposition 42 (Position of a language) *Given a Petri net language,*

- *If the labelling function is the identity and every marking is a terminal marking, one can decide whether it is regular [VAL 81] and whether it is algebraic [SCH 92].*
- *If furthermore a set of terminal markings is specified, one can still decide whether it is regular [LAM 92].*

From the point of view of position of Petri net languages, the model of recursive nets unifies Petri nets and algebraic grammars. Indeed, the family of languages of recursive nets strictly includes the union of Petri net and algebraic languages and as for these families one can decide the membership and the emptyness problem. However unlike these families, the intersection of a recursive net language and a regular language is not necessarily a recursive net language.

References

[ABD 96] P.A. Abdulla, K. Ĉerāns, Jonsson B. and Yih-Kuen T. General decidability theorems for infinite-state systems. In *Proc.* 11^{th} *IEEE Symp. Logic In Computer Science (LICS'96)*, LNCS, pages 313–321, New Brunswick, NJ, USA, Juillet 1996.

[AHO 74] Alfred V. Aho, John E. Hopcroft and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, 1974.

[ARN 76] A. Arnold and M. Latteux. Vector addition systems and semi-Dick language. Rapport de Recherche 78, Laboratoire de Calcul, Université des Sciences et Techniques de Lille, Décembre 1976.

[AUT 87] J.M. Autebert. *Langages algébriques*. Etudes et recherches en informatique. Masson, 1987.

[DUF 98a] C. Dufourd. Réseaux de Petri avec reset/transfert : Décidabilité et indécidabilité. Thèse d'Université de l'ENS de Cachan. Laboratoire Spécification et Vérification, Octobre 1998.

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

[ESP 94] J. Esparza and M. Nielsen. Decidability issues for Petri nets - A survey. *Bulletin of the EATCS*, 52:245–262, 1994.

[ESP 98] J. Esparza. Decidability and complexity of Petri net problems - an introduction. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer Verlag, 1998.

[FIN 98] A. Finkel and P. Schnoebelen. Fundamental structures in well-structured infinite transition. In *Proc.* 3^{rd} *Latin American Theoretical Informatics Symposium (LATIN'98)*, volume 1380 of *LNCS*, pages 102–118, Campinas, Brésil, Avril 1998.

[FRU 86] D. Frutos Escrig. Decidability of home states in place transition systems. Rapport interne, Dpto. Informatica y Automatica., Univ. Complutense de Madrid, 1986.

[FRU 89] D. Frutos Escrig and C. Johnen. Decidability of home space property. Rapport LRI-503, Laboratoire de Recherche en Informatique, Univ. de Paris-Sud, Orsay, 1989.

[HAC 75] M. Hack. *Decidability Questions for Petri Nets*. PhD thesis, M.I.T., Cambridge, MA, Décembre 1975. publié comme rapport technique 161, Lab. for Computer Science, Juin 1976.

[HAD 99a] S. Haddad and D. Poitrenaud. Decidability and undecidability results for recursive Petri nets. Rapport de Recherche 019, LIP6, Paris VI University, Paris, France, Septembre 1999.

[HAD 99b] S. Haddad and D. Poitrenaud. Theoretical aspects of recursive Petri nets. In *Proc.* 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, Juin 1999. Springer Verlag.

[HAD 00] S. Haddad and D. Poitrenaud. Modelling and analyzing systems with recursive Petri nets. In *Proc. of the Workshop on Discrete Event Systems - Analysis and Control*, pages 449–458, Gand, Belgique, August 2000. Kluwer Academics Publishers.

[HAD 07] S. Haddad and D. Poitrenaud. Recursive Petri nets – Theory and application to discrete event systems. *Acta Informatica*, 44(7-8):463–508, December 2007.

[HAU 90] D. Hauschildt. Semilinearity of the reachability set is decidable for Petri nets. Rapport FBI-HH-B-146/90, Université de Hambourg, 1990.

[HOP 69] J. E. Hopcroft and J. D. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, Reading, 1969.

[HOP 79] J. Hopcroft and J-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 5:135–159, 1979.

[JAN 79] M. Jantzen. On the hierarchy of Petri net languages. *R.A.I.R.O. Informatique Theorique*, 13:19–30, 1979.

[KAR 69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

[KOS 82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc.* 14^{th} *ACM Symp. Theory of Computing (STOC'82)*, pages 267–281, San Francisco, CA, Mai 1982.

[LAM 92] J.L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99:79–104, 1992.

[LIP 76] R. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[MAY 81] E. Mayr and A. Meyer. The complexity of the finite containment problem for Petri nets. *Journ. Assoc. Comput. Mach.*, 28:561–576, 1981.

[MAY 84] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, 13:441–460, 1984.

[PAP 94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Mass., 1994.

[PET 81] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice Hall, 1981.

[RAC 78] C. Rackoff. The covering and boudedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[REI 95] K. Reinhardt. Reachability in Petri nets with inhibitor arcs. Unpublished manuscript reachable via www-fs.informatik.uni-tuebingen.de/∼reinhard, 1995.

[REU 89] C. Reutenauer. *Aspects mathématiques des réseaux de Petri*. Etudes et recherches en informatique. Masson, 1989.

[SCH 92] S. Schwer. The context-freeness of the languages associated with vector addition systems is decidable. *Theoretical Computer Science*, 98:199–247, 1992.

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

[VAL 81] R. Valk and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 3(23):299–325, 1981.

[VAL 85] R. Valk and M. Jantzen. The residue of vectors sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21:643–674, 1985.

[YEN 92] H-C. Yen. A unified approach for deciding the existence of certain Petri net paths. *Information and Computation*, 96:119–137, 1992.

Index