Reachability Analysis of (Timed) Petri Nets Using Real Arithmetic

Béatrice Bérard and Laurent Fribourg

LSV – ENS de Cachan/CNRS - 61 av. Pdt. Wilson - 94235 Cachan - France
email: {berard,fribourg}@lsv.ens-cachan.fr

Abstract. In this paper, we address the issue of reachability analysis for Petri nets, viewed as automata with counters. We show that exact reachability analysis can be achieved by treating Petri nets integer variables (counters) as real-valued variables, and using Fourier-Motzkin procedure instead of Presburger elimination procedure. As a consequence, one can safely analyse Petri nets with performant tools, e.g. HyTech, originally designed for analysing automata with real-valued variables (clocks). We also investigate the use of *meta-transitions* (iterative application of a transition in a single step) and give sufficient conditions ensuring an exact computation in this case. Experimental results with HyTech show an impressive speed-up with respect to previous experiences performed with a Presburger arithmetic solver. The method extends for analysing Petri nets with inhibitors and with timing constraints, but difficulties arise for the treatment of meta-transitions in the latter case.

1 Introduction

Reachability analysis with Presburger arithmetic. As surveyed by Esparza [18], there are several classes of Petri nets whose set of reachable markings can be described as a formula of Presburger arithmetic (i.e. a linear arithmetic formula over integers). In practice, it turns out that iterative symbolic computation of reachable markings, when it converges, often yields a set $post^*$ expressible in Presburger arithmetic, even for Petri nets which do not belong to the classes mentioned above [20,29]. Such a practical result was also observed in [9,32] in the case of infinite-state systems with integers, called "extended finite state automata (EFSM)". These observations have led independently several researchers to implement procedures, based on Presburger arithmetic solvers, for trying and compute the reachability sets of extended forms of automata [9,20,32]. Note that there is generally no guarantee that the computation terminates. Moreover, the complexity of Presburger arithmetic is extremely high, a major obstacle for dealing with more than small-size examples. In this paper we will try to alleviate this complexity problem.

Can real arithmetic be used instead? It is well known that real arithmetic is more tractable than integer arithmetic: for instance, checking the existence of a solution over $\mathbb{R}$ for a system of linear inequations is polynomial while it is NP-hard over $\mathbb{N}$ [31]. In general resolution over $\mathbb{R}$ leads to an *upper approximation* of the set of integer solutions. However Shostack [33] noticed that, for a particular class of formulas (made of constraints of the form $x + c \leq y$, $x \leq c$ or $c \leq x$), elimination of variables over $\mathbb{R}$ using Fourier-Motzkin procedure is *exact*, i.e. produces the same result as elimination of variables over $\mathbb{N}$. For this class, it is safe to focus on "polyhedric" constraints (i.e. sets of linear inequalities), and discard the cumbersome divisibility constraints that are *a priori* needed when dealing with integers.

Contribution of the paper. Following Shostack, we show here that reachability analysis of Petri nets can be done by treating integer variables as real-valued variables, and eliminating them with Fourier-Motzkin. As a consequence, given a Petri net and an initial marking, iterative computation of the successor markings with Fourier-Motzkin yields the exact reachability set $post^*$, when it terminates. The same is true for backward computation of predecessors pre^*.
The idea of solving over reals rather than integers was recently used by Delzanno and Podelski in [17]. They deal with EFSM rather than Petri nets, and verify not only safety properties but also liveness properties. On the other hand, in contrast with us, they do not obtain in general an exact result for the reachability sets but only an overapproximation. Besides, we treat here the case of "(cycle) meta-transitions" (see [7,35]). A meta-transition simulates the iteration of the same transition applied an arbitrary number μ of times, often entailing a better convergence for $post^*$ and pre^*. Fourier-Motzkin elimination (of μ) does not give any longer an exact result with meta-transitions, but only an upper approximation. We still give a sufficient condition on the intermediate formulas generated by the procedure, that guarantees an exact computation.
Our method extends to Petri nets with inhibitors and Petri nets with timing constraints although problems arise with meta-transitions in the latter case. We experimented the procedure on examples of various sizes, including a part of a communication protocol (PNCSA, [22]) in which a deadlock was discovered.

Plan of the paper. After recalling some basic definitions in §2, we give results about ordinary Petri nets in §3, about Petri nets with meta-transitions in §4, and about timed nets in §5; we conclude in §6.

2 Preliminaries

2.1 Fourier-Motzkin variable elimination

This presentation is adapted from [25]. An *atomic constraint* is a linear inequality of the form $\Sigma_{i=1}^{n} a_i x_i \geq c$ or equality $\Sigma_{i=1}^{n} a_i x_i = c$ with coefficients in $\mathbb{Z}$. A *constraint* is a conjunction of atomic constraints. We assume every atomic constraint is written in a simplified form so that no variable appears twice in the same atomic constraint. Let $vars(\phi)$ be the set of variables appearing in constraint ϕ. For each variable $x \in vars(\phi)$ we can partition ϕ in the following way. Let ϕ_x^0 be the set of atomic constraints of ϕ which do not contain any occurrence of x. Let ϕ_x^+ (resp ϕ_x^-) be the set of inequalities of ϕ equivalent to $ax \leq e$ (resp. $e \leq bx$), where e is a linear expression not involving x. The projection algorithm shown below eliminates a variable x from constraint ϕ and returns constraint $\psi \leftrightarrow \exists x \phi$ using Fourier-Motzkin elimination (and replacement of equals).

project (ϕ, x)
`if` $\phi \leftrightarrow \phi' \wedge x = e$ `then`
% Replacement of equals
  $\psi := \phi'$ with every occurrence of x replaced by e
`else`
% Fourier-Motzkin elimination
  $\psi := \phi_x^0$
  `foreach` $A \in \phi_x^+$ with $A \leftrightarrow ax \leq e^+$
    `foreach` $B \in \phi_x^-$ with $B \leftrightarrow e^- \leq bx$
      $\psi := \psi \wedge ae^- \leq be^+$
    `endforeach`
  `endforeach`
`endif`
`return` ψ;

Formula ψ is only an upper approximation of the set of *integer* solutions because the algorithm assumes that there is always a solution for x to $ae^- \leq abx \leq be^+$, as soon as $ae^- \leq be^+$. This is true over $\mathbb{R}$ but not over $\mathbb{N}$. Note however that if $a = 1$ or $b = 1$, such an integer solution always exists. Therefore, in the particular case where, in ϕ, all the coefficients of x in lower bounds on x are 1, or all the coefficients of x in upper bounds on x are 1, Fourier-Motzkin is *exact* over $\mathbb{N}$: formula ψ characterizes the set of integer solutions (see [30]). This observation will be used in §4.
In the following, elimination will be done implicitly via a projection algorithm over $\mathbb{R}$, such as Fourier-Motzkin. Experimental results given throughout the paper were obtained using HyTech[1][26], which has a built-in procedure of projection over reals (taken from Halbwachs' polyhedral manipulation library [24]).

2.2 Petri Nets

A *net* N is a tuple $\langle X, R, \alpha, \beta \rangle$, where
- $X = \{x_1, \ldots, x_m\}$ is a finite set of counters[2],
- R is a finite set of transitions,
- $\alpha : X \times R \mapsto \mathbb{N}$ and $\beta : R \times X \mapsto \mathbb{N}$ are valuation functions.

The value $\alpha(x_i, r)$ represents the weight of an *input-arc* from *input-counter* x_i to r, and $\beta(r, x_i)$ the weight of an *output-arc* from r to *output-counter* x_i. In the

[1] on a SUN station ULTRA-1 with 64 Megabytes of RAM memory
[2] The usual terminology is *places*, but it is convenient here to refer to *counters*.

figures, counters are drawn as circles, transitions as bars. An arc is represented as a weighted arrow, only if it has a non zero weight. The weights of the drawn arcs are implicitly 1. With each transition $r \in R$ in the net is associated a couple of vectors $(\overline{c}_r, \overline{a}_r)$, where the i-th component c_r^i of $\overline{c}_r$ is $\beta(r, x_i) - \alpha(x_i, r)$, and the i-th component a_r^i of $\overline{a}_r$ is $\alpha(x_i, r)$. A *marking* is a mapping from X to $\mathbb{N}$. A set of markings is represented as a formula $\varphi(\overline{x})$ interpreted over $\mathbb{N}$, where $\overline{x}$ is the vector of variables $x_1, \cdots, x_m$. A *Petri net* is a pair (N, φ_{init}) where N is a net, and φ_{init} a formula, called "initial formula", characterizing the set of initial markings. Formula φ_{init} is generally of the form $\overline{x} = \overline{x}_{init}$, with $\overline{x}_{init} \in \mathbb{N}^m$. We also consider cases where the initial marking is *parametric*: some counters are assigned parameters instead of constant values. Henceforth we will assume given a set $P = \{p_1, p_2, \ldots\}$ of parameters which are additional variables, the value of which are left unchanged by transitions. A parametric initial formula $\varphi_{init}(\overline{x}, \overline{p})$ is of the form $\overline{x} = \overline{x}_{init}$ with $\overline{x}_{init} \in (\mathbb{N} \cup P)^m$ (and possibly some additional constraints over $\overline{p}$). For a transition $r \in R$ corresponding to $(\overline{c}_r, \overline{a}_r)$, the *reachability relation via* r, written $\xrightarrow{r}$, is defined by:

$$\overline{x} \xrightarrow{r} \overline{x}' \quad \Leftrightarrow \quad \overline{x}' = \overline{x} + \overline{c}_r \wedge \overline{x} \geq \overline{a}_r .$$

Given a net $N : \langle X, R, \alpha, \beta \rangle$, the *reachability relation via* N, written $\xrightarrow{N}$, is $\bigcup_{r \in R} \xrightarrow{r}$. (This means: $\overline{x} \xrightarrow{N} \overline{x}'$ iff $\exists r \in R \;\; \overline{x} \xrightarrow{r} \overline{x}'$.) Without loss of understanding in the following, we will use R and $\xrightarrow{R}$ in place of N and $\xrightarrow{N}$ respectively. For lack of space we do not explain how to treat Petri nets with *inhibitors* ("0-tests") in this paper, but all our definitions and results easily extend to them.

Example 1. The Petri net in figure 1 corresponds to the "swimming-pool" example from M. Latteux (see [8,16]). All weights are equal to 1. Transition r_1 (incoming guest) is enabled if $x_6 \geq 1$, when there is at least one free cabine. When r_1 is fired, one token is removed from counter x_6 and added into counter x_1, meaning the beginning of undressing. The initial marking $\overline{x}_{init}$ involves two parameters p_1 and p_2 (assumed to be greater than or equal to 1), which represent the number of available cabins and baskets respectively. We have $\varphi_{init} :\; x_1 = x_2 = x_3 = x_4 = x_5 = 0 \;\wedge\; x_6 = p_1 \wedge x_7 = p_2 \wedge p_1 \geq 1 \wedge p_2 \geq 1$. The reachability relation via r_1 is: $\quad \overline{x} \xrightarrow{r_1} \overline{x}'$ iff

$x'_1 = x_1 + 1 \wedge x'_6 = x_6 - 1 \wedge x_6 \geq 1 \wedge x'_2 = x_2 \wedge x'_3 = x_3 \wedge x'_4 = x_4 \wedge x'_5 = x_5$.

Relations for transitions r_2 to r_6 are similar. For the sake of brevity, all the equations of the form $x'_i = x_i$ (counters unchanged), will be omitted in the next examples. Also recall that inequations of the form $x_i \geq 0$ are always implicit.

3 Reachability analysis in Petri nets

Given a formula φ and a transition r, the classical notion of successor (resp. predecessor), characterizing the markings reachable via r (resp. r^{-1}) from a marking satisfying φ, is defined as follows.

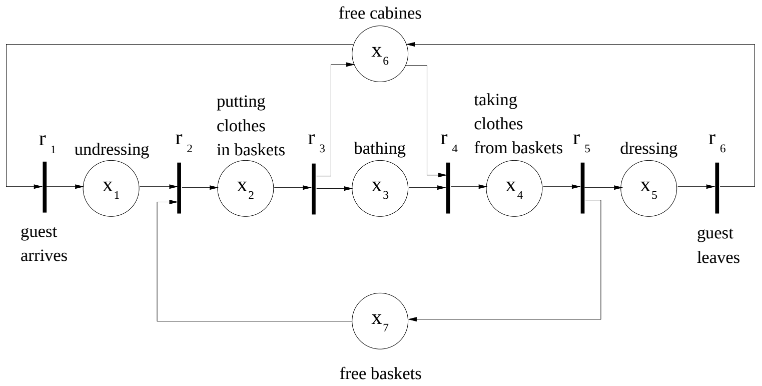

Fig. 1. A Petri net for the swimming pool example

The *successor of* φ *via* r, written $post_r(\varphi)$, is the result of eliminating $\overline{x}'$ over $\mathbb{N}$ from: $\quad \exists \overline{x}' \ \varphi(\overline{x}') \wedge \overline{x}' \xrightarrow{r} \overline{x}$.
The *predecessor of* φ *via* r, written $pre_r(\varphi)$, is the result of eliminating $\overline{x}'$ over $\mathbb{N}$ from: $\quad \exists \overline{x}' \ \varphi(\overline{x}') \wedge \overline{x} \xrightarrow{r} \overline{x}'$.
Since $\overline{x} \xrightarrow{r} \overline{x}'$ means $\overline{x}' = \overline{x} + \overline{c}_r \wedge \overline{x} \geq \overline{a}_r$, we have (by replacement of equals):

$$\begin{array}{lll} post_r(\varphi) & \Leftrightarrow & \varphi(\overline{x} - \overline{c}_r) \wedge \overline{x} \geq \overline{a}_r + \overline{c}_r\,. \\ pre_r(\varphi) & \Leftrightarrow & \varphi(\overline{x} + \overline{c}_r) \wedge \overline{x} \geq \overline{a}_r\,. \end{array}$$

As usual, given a set R of transitions, $post_R(\varphi)$ is $\bigcup_{r \in R} post_r(\varphi)$ (the subscript R is sometimes omitted when it stands for the set of all transitions of the Petri net) and the iterated operator $post^j$ for a formula φ is defined recursively by:
$\quad post^0(\varphi) = \varphi, \ \ post^{j+1}(\varphi) = post(post^j(\varphi))$.
The infinite union $\bigcup_{j \geq 0} post^j(\varphi)$, written $post^*(\varphi)$, characterizes the set of all the markings reachable from φ through a finite sequence of transitions. We say that the iterative computation of $post^*$ *stabilizes* or *terminates at step* k when k is the first index such that $post^{k+1}(\varphi) \subseteq \bigcup_{0 \leq j \leq k} post^j(\varphi)$. In this case $post^*(\varphi) = \bigcup_{0 \leq j \leq k} post^j(\varphi)$. One defines similarly $pre^j(\varphi)$ and $pre^*(\varphi)$.

We are interested in (dis)proving that all the reachable markings of a Petri net satisfy a certain "safety" property, e.g. deadlock-freeness. This will be done by considering a formula φ_{safe}, which characterizes the set of all the safe markings, and checking: $\quad post^*(\varphi_{init}) \cap \neg\varphi_{safe} = \emptyset \quad$ or $\quad pre^*(\neg\varphi_{safe}) \cap \varphi_{init} = \emptyset$.
In the examples hereafter, we will (dis)prove the absence of deadlock. The property $\varphi_{safe}(\overline{x})$, expressing that $\overline{x}$ enables at least one transition, is: $\bigvee_{r \in R} \overline{x} \geq \overline{a}_r$. Its negation $\neg\varphi_{safe}$, characterizing the set of all the deadlock markings, is of the form $\bigvee_{k \in K} \varphi_{dead_k}$ where K is a finite set of indices and φ_{dead_k} characterizes a subset of deadlock markings. There is a deadlock iff :
$\quad post^*(\varphi_{init}) \cap \varphi_{dead_k} \neq \emptyset \quad$ or $\quad pre^*(\varphi_{dead_k}) \cap \varphi_{init} \neq \emptyset, \quad$ for some $k \in K$.

Definition 2. A formula $\varphi(\overline{x})$ is a *counter region* iff it is a conjunction of inequalities of the form:

$$x_i + d \geq x_j, \quad x_i \geq x_j + d, \quad x_i \geq d, \quad d \geq x_i, \qquad \text{with } x_i, x_j \in X \cup P,\ d \in \mathbb{N}.$$

We say that a formula $\varphi(\overline{x})$ *belongs to counter class* **C** iff φ is a finite disjunction of counter regions.

Proposition 3. *Let φ be a formula of* **C**. *Then:*

1. *Variable elimination with Fourier-Motzkin is exact for computing $post_R(\varphi)$.*
2. *$post_R(\varphi)$ belongs to* **C**.
3. *When it terminates, the iterative computation of $post^j_R(\varphi)$ (using Fourier-Motzkin) yields a formula $post^*_R(\varphi)$ belonging to* **C**.

Proof. Statement 1 follows from the fact that elimination of $\overline{x}'$ in $\exists \overline{x}'\, \varphi(\overline{x}') \wedge \overline{x}' \xrightarrow{r} \overline{x}$ is merely an operation of replacement of equals, and does not use the integer assumption for $\overline{x}'$. Besides the resulting formula $\varphi(\overline{x} - \overline{c}_r) \wedge \overline{x} \geq \overline{a}_r + \overline{c}_r$ belongs to class **C** since $\varphi \in$ **C**: this is because replacing x_i (resp. x_j) with $x_i - c^i_r$ (resp. $x_j - c^j_r$) in a constraint of the form $x_i \geq x_j + d$, $x_i + d \geq x_j$, $x_i \geq d$, or $d \geq x_i$ yields a constraint of the same form. Therefore statement 2 holds. Statement 3 follows from statements 1 and 2.

The proposition also holds with *pre* instead of *post*. Since φ_{init} belongs to **C**, it follows from the proposition that computation of $post^*$ yields a formula of **C** when it terminates.

Example 4. This example is taken from [10] (cf. [36]) and illustrated by the Petri net of figure 2. It models an automated manufacturing system with four

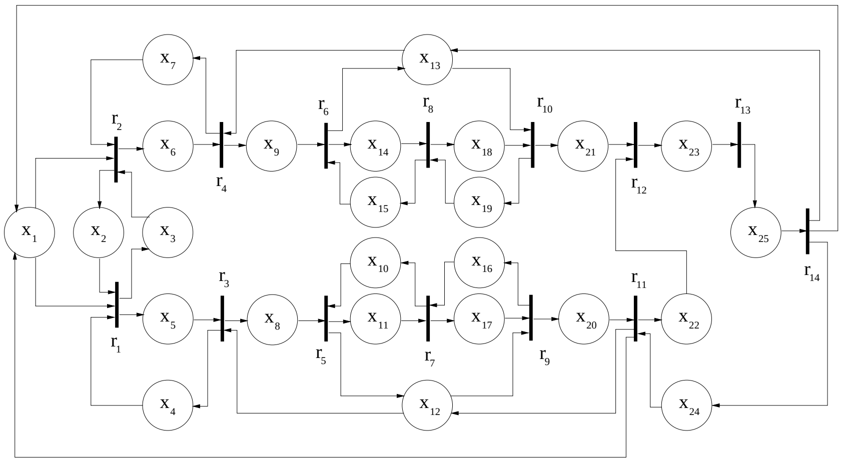

Fig. 2. A manufacturing system

machines, two robots, two buffers (x_{10} and x_{15}) and an assembly cell. The initial marking $\overline{x}_{init}$ is such that: $x_1 = p$ for some nonnegative parameter p, $x_2 = x_4 = x_7 = x_{12} = x_{13} = x_{16} = x_{19} = x_{24} = 1$, $x_{10} = x_{15} = 3$ (buffers have capacity 3). All other counters are empty (that is, all other variables are 0). The task is to discover for which values of parameter p the system may end up in a deadlock.
Previous works: In [36], deadlock-freeness is shown only for $1 \leq p \leq 4$. In [10], deadlock-freeness is proved using some mixed integer programming techniques for $1 \leq p \leq 8$. A path leading to a deadlock is then generated for $p = 9$. In [20] the exact reachability set has been automatically computed using variable elimination over $\mathbb{N}$, showing that a deadlock is reached from $\overline{x}_{init}$ if and only if $p \geq 9$, but the computation takes several hours (2.5 hours on a SUN station ULTRA-1).
New results: The computation of $post^*(\varphi_{init})$ with HyTech stabilizes after 81 iterations, and takes only a couple of minutes (200 seconds). The output set $post^*$ is a disjunction of 560 counter regions. More precisely, each region is of the form $(p = x_1 + d_0 \wedge x_1 \geq d_1 \wedge \bigwedge_{i=2}^{25} x_i = d_i)$ or $(p = d_0 \wedge x_1 = d_1 \wedge \bigwedge_{i=2}^{25} x_i = d_i)$, with $d_i \in \{0, 1, \cdots, 12\}$ for all $0 \leq i \leq 25$. The intersection of $post^*$ with the formula characterizing deadlock markings gives $p = x_1 + 9 \;\wedge\; x_1 \geq 0 \wedge x_{11} = 3 \wedge x_{15} = 2 \wedge x_3 = x_4 = x_6 = x_8 = x_{14} = x_{17} = x_{18} = x_{21} = x_{24} = 1$, all the other variables being null. We thus retrieve the necessary and sufficient condition of deadlock-freeness ($p \leq 8$). Note the impressive computation time speed-up gained here w.r.t. [20].

4 Petri Nets with Meta-Transitions

4.1 Meta-transitions

The computation of $post^*$ (or pre^*), performed in a brute manner as explained above, often converges slowly or does not terminate at all. A classical way to speed-up the convergence and improve the termination of $post^*$ is to use "(cycle) meta-transitions" ([7,35]), i.e. repeated application of a transition r an arbitrary number μ of times in a single step. Let us define formally this notion in our context. Given $\mu \in \mathbb{N} \setminus \{0\}$, the *$\mu$-reachability* relation via transition $r : (\overline{c}_r, \overline{a}_r)$, written $\xrightarrow{r^\mu}$, is defined by:

$$\overline{x} \xrightarrow{r^\mu} \overline{x}' \;\Leftrightarrow\; \overline{x} \xrightarrow{r} \overline{x}' \qquad \text{if } \mu = 1,$$
$$\overline{x} \xrightarrow{r^\mu} \overline{x}' \;\Leftrightarrow\; \exists \overline{x}_1 \;\; \overline{x} \xrightarrow{r} \overline{x}_1 \xrightarrow{r^{\mu-1}} \overline{x}' \qquad \text{if } \mu \geq 2.$$

The *transitive closure* of $\xrightarrow{r}$, written $\xrightarrow{r^+}$, is defined by:

$$\overline{x} \xrightarrow{r^+} \overline{x}' \;\Leftrightarrow\; \exists \mu \in \mathbb{N} \setminus \{0\} \;\; \overline{x} \xrightarrow{r^\mu} \overline{x}'$$

Proposition 5. *[20] Given a transition $r : (\overline{c}_r, \overline{a}_r)$, we have:*

$$\overline{x} \xrightarrow{r^\mu} \overline{x}' \quad \textit{iff} \quad \overline{x}' = \overline{x} + \mu.\overline{c}_r \;\wedge\; \overline{x} + (\mu - 1).\overline{c}'_r \geq \overline{a}_r,$$

where $\overline{c}'_r$ is obtained from $\overline{c}_r$ by replacing all the positive components with 0.

Proof. $\quad \overline{x} \xrightarrow{r^\mu} \overline{x}'$

$\Leftrightarrow \qquad \exists \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_{\mu-1} \quad \overline{x} \xrightarrow{r} \overline{x}_1 \wedge \overline{x}_1 \xrightarrow{r} \overline{x}_2 \wedge \cdots \wedge \overline{x}_{\mu-1} \xrightarrow{r} \overline{x}'$
$\Leftrightarrow \qquad \overline{x}' = \overline{x} + \mu.\overline{c}_r \quad \wedge \quad (\overline{x} \geq \overline{a}_r \ \wedge \overline{x} + 1.\overline{c}_r \geq \overline{a}_r \wedge \cdots \ \wedge \overline{x} + (\mu - 1).\overline{c}_r \geq \overline{a}_r)$.
Conjunction $(\overline{x} \geq \overline{a}_r \ \wedge \overline{x} + 1.\overline{c}_r \geq \overline{a}_r \wedge \cdots \ \wedge \overline{x} + (\mu - 1).\overline{c}_r \geq \overline{a}_r)$ simplifies to $\overline{x} + (\mu - 1).\overline{c}'_r \geq \overline{a}_r$, with $\overline{c}'_r$ as defined above (see [20] for details).

Example 6. Consider transition r_1 of the swimming-pool example:
$\overline{x} \xrightarrow{r_1} \overline{x}'$ iff $\quad x'_1 = x_1 + 1 \wedge x'_6 = x_6 - 1 \wedge x_6 \geq 1$.
From $\overline{c}_r = (1, 0, 0, 0, 0, -1, 0)$, we obtain $\overline{c}'_r = (0, 0, 0, 0, 0, -1, 0)$. Besides $\overline{a}_r$ is $(0, 0, 0, 0, 0, 1, 0)$, hence: $\overline{x} \xrightarrow{r_1^\mu} \overline{x}'$ iff $\quad x'_1 = x_1 + \mu \wedge x'_6 = x_6 - \mu \wedge x_6 - (\mu - 1) \geq 1$, i.e. $\quad x'_1 = x_1 + \mu \wedge x'_6 = x_6 - \mu \wedge x_6 \geq \mu$.

Definition 7. The *μ-successor of φ via r*, denoted $post_{r^\mu}(\varphi)$, is such that:
$$post_{r^\mu}(\varphi) \quad \Leftrightarrow \quad \exists \overline{x}' \ \varphi(\overline{x}') \wedge \overline{x}' \xrightarrow{r^\mu} \overline{x}.$$
The *meta-successor of φ via r*, written $Mpost_r(\varphi)$, is such that
$$Mpost_r(\varphi) \quad \Leftrightarrow \quad \exists \overline{x}' \ \varphi(\overline{x}') \wedge \overline{x}' \xrightarrow{r^+} \overline{x} \qquad \Leftrightarrow \quad \exists \mu \in \mathbb{N} \setminus \{0\} \ \ post_{r^\mu}(\varphi).$$

It is straightforward to show (using proposition 5):

$$post_{r^\mu}(\varphi) \quad \Leftrightarrow \quad \varphi(\overline{x} - \mu.\overline{c}_r) \wedge \overline{x} + \mu(\overline{c}'_r - \overline{c}_r) \geq \overline{a}_r + \overline{c}'_r.$$

We define also: $post_{R^\mu}(\varphi) = \bigcup_{r \in R} post_{r^\mu}(\varphi)$, and $Mpost_R(\varphi) = \bigcup_{r \in R} Mpost_r(\varphi)$.
Since only transitions of R are used in $Mpost_R$, we have: $\qquad Mpost^*_R = post^*_R$.

If one regards μ not as a natural number but as a *real*, and eliminate μ via Fourier-Motzkin from $\exists \mu \geq 1 \ post_{r^\mu}(\varphi)$, one does not generally get an exact result, but only an upper approximation of $Mpost_r(\varphi)$. However, in the special case where all the coefficients of μ in $post_{r^\mu}(\varphi)$ for lower bounds of μ are 1, Fourier-Motzkin elimination is guaranteed to be exact (see §2), and yields a formula equivalent to $Mpost_r(\varphi)$. The same result holds if all the nonull coefficients of μ in upperbounds of μ are 1. This suggests to compute an upper approximation, denoted $Upost_R(\varphi)$, of $Mpost_R(\varphi)$ as follows.

Proposition 8. *Given a formula φ, let $Upost_R(\varphi)$ be the result of eliminating μ via Fourier-Motzkin from $\exists \mu \geq 1 \ post_{R^\mu}(\varphi)$. We have:*

$$Upost^*_R(\varphi) \supseteq Mpost^*_R(\varphi) = post^*_R(\varphi).$$

Besides, if iterative computation of $Upost^j_R(\varphi)$ stabilizes at step k, and all the coefficients of μ on lower bounds (resp. upper bounds) of μ in $post_{r^\mu}(Upost^j_R(\varphi))$ are 1, for every $0 \leq j \leq k$ and every $r \in R$, then:

$$Upost^*_R(\varphi) = Mpost^*_R(\varphi) = post^*_R(\varphi).$$

One defines similarly the notions $Mpre_r, Upre_r, Mpre_R, Upre_R$. The counterpart of the above proposition holds. The examples given at the end of the section show that iterative computation of $Upre_R$ is a practical and efficient way of computing an approximation of pre^*_R. Inspection of μ coefficients within intermediate formulas $pre_{r^\mu}(Upre^j_R(\varphi))$ will guarantee additionally that $Upre^*_R = pre^*_R$.

4.2 Fused transitions

In the rest of this section we consider, instead of the original set $R = \{r_1, \cdots, r_n\}$ of transitions of the given Petri net, a new set R' obtained from R by adding sequences of transitions, called *fused transitions*, of the form $r_{i_1} \cdots r_{i_k}$. Fused transitions can be derived in practice through a process of "decomposition" of R (see [21]), which uses a generalized form of fusion rules originally proposed by Berthelot [5] for transforming Petri nets.

The reachability relation via $r_{i_1} \cdots r_{i_k}$, written $\xrightarrow{r_{i_1} \cdots r_{i_k}}$, is defined by:

$$\overline{x} \xrightarrow{r_{i_1} \cdots r_{i_k}} \overline{x}' \quad \text{iff} \quad \exists \overline{x}_1, \cdots, \overline{x}_{k-1} \quad \overline{x} \xrightarrow{r_{i_1}} \overline{x}_1 \xrightarrow{r_{i_2}} \cdots \xrightarrow{r_{i_{k-1}}} \overline{x}_{k-1} \xrightarrow{r_{i_k}} \overline{x}'.$$

Alternatively, we can consider a fused transition $\sigma = r_{i_1} \cdots r_{i_k}$ as a transition associated with a couple $(\overline{c}_\sigma, \overline{a}_\sigma)$. Writing ε for the empty sequence, we have:

Definition 9. Given a fused transition σ, the *reachability relation via* σ, written $\xrightarrow{\sigma}$, is defined by $\quad \overline{x} \xrightarrow{\sigma} \overline{x}' \quad \Leftrightarrow \quad \overline{x}' = \overline{x} + \overline{c}_\sigma \wedge \overline{x} \geq \overline{a}_\sigma,$

with $\overline{c}_\sigma$ and $\overline{a}_\sigma$ recursively defined as:

$$\begin{array}{lll} \overline{c}_\varepsilon = \overline{0} & \quad & \overline{c}_{r.\sigma} = \overline{c}_r + \overline{c}_\sigma \\ \overline{a}_\varepsilon = \overline{0} & \quad & \overline{a}_{r.\sigma} = max(\overline{a}_r, \overline{a}_\sigma - \overline{c}_r) \end{array}$$

Since R' is obtained from R by adding sequences of transitions of R, we have: $post^*_{R'} = post^*_R \quad$ and $\quad pre^*_{R'} = pre^*_R$.

Example 10. Consider the transitions r_1, r_2 of the swimming-pool example:

$\overline{x} \xrightarrow{r_1} \overline{x}' \quad$ iff $\quad x'_1 = x_1 + 1 \wedge x'_6 = x_6 - 1 \wedge x_6 \geq 1.$

$\overline{x} \xrightarrow{r_2} \overline{x}' \quad$ iff $\quad x'_1 = x_1 - 1 \wedge x'_2 = x_2 + 1 \wedge x'_7 = x_7 - 1 \wedge x_1 \geq 1 \wedge x_7 \geq 1.$

The fused transition $r_1 r_2$ is characterized by $(\overline{c}_{r_1 r_2}, \overline{a}_{r_1 r_2})$ where $\overline{c}_{r_1 r_2}$ is $(0, 1, 0, 0, 0, -1, -1)$ and $\overline{a}_{r_1 r_2}$ is $(0, 0, 0, 0, 0, 1, 1)$. We have:

$\overline{x} \xrightarrow{r_1 r_2} \overline{x}' \quad$ iff $\quad x'_2 = x_2 + 1 \wedge x'_6 = x_6 - 1 \wedge x'_7 = x_7 - 1 \wedge x_6 \geq 1 \wedge x_7 \geq 1.$

4.3 Examples

In the following examples we consider an extended set R' obtained from R by adding fused transitions. We compute $Upre^*_{R'}$ and show that: $Upre^*_{R'} = Mpre^*_{R'}$ $(= pre^*_{R'} = pre^*_R)$. The set R' has been obtained by a preliminary decomposition of R (see [21]) up to level k, i.e. until k distinct fused transitions have been generated. The computation time of the decomposition process is neglectable w.r.t. computation time of $post^*$ (or pre^*). Note that the choice of the appropriate value k is the result of a compromise: the bigger k is, the better the convergence of $post^*$ is likely to be, but the space search may explode for too large values of k. In each of the examples below, we first had to try a couple of values for k before obtaining a convenient one. One can also improve the convergence by dropping certain "troublesome" fused transitions, but this issue is beyond the scope of the paper.[3]

[3] The decomposition process also infers an *ordering* of application among the set of ordinary and fused transitions, but this information is not exploited here.

Example 11. Consider the set of transitions $R = \{r_1, r_2, \cdots, r_6\}$ for the swimming pool Petri net. The Petri net derived by decomposition has a set R' with the following 6 fused transitions $\{r_2r_3,\ r_1r_2r_3,\ r_6r_4,\ r_4r_5,\ r_5r_6r_4,\ r_1r_2r_3r_4r_5\}$ added to R. The deadlock formula is of the form $\varphi_{dead_1} \vee \varphi_{dead_2}$ with:

$\varphi_{dead_1} : x_2 = x_4 = x_5 = x_6 = x_7 = 0, \quad \varphi_{dead_2} : x_1 = x_2 = x_4 = x_5 = x_6 = 0.$

The iterated computation of $Upre_{R'}$ applied to φ_{dead_1} terminates after 22 steps (and takes 50 seconds). Besides for all $j \leq 22$, one can check that all the nonnull coefficients of μ in $pre_{R'^{\mu}}(Upre^{j}_{R'}(\varphi_{dead_1}))$ are equal to 1. Therefore there is no upper approximation: $Upre^{*}_{R'}(\varphi_{dead_1}) = pre^{*}_{R}(\varphi_{dead_1})$. It is set of 10 constraints:

$\{x_2 = x_4 = x_5 = x_6 = x_7 = 0, \quad x_2 = x_4 = x_5 = x_7 = 0 \wedge x_6 \geq 1,$
$x_2 = x_4 = x_7 = 0 \wedge x_5 \geq 1, \quad x_2 \geq 1, \quad x_1 \geq 1 \wedge x_7 \geq 1, \quad x_6 \geq 1 \wedge x_7 \geq 1,$
$x_4 \geq 1, \quad x_3 \geq 1 \wedge x_6 \geq 1, \quad x_5 \geq 1 \wedge x_7 \geq 1, \quad x_3 \geq 1 \wedge x_5 \geq 1\}$

One of these constraints, $x_6 \geq 1 \wedge x_7 \geq 1$, has ϕ_{init} as an instance. It follows that the system always has a deadlock, for any values of parameters p_1 and p_2 (greater than or equal to 1). It is often interesting to try and keep track of a path linking the initial marking to a deadlock, but this requires to store additional information during the construction of $Upre^*$ at a cost generally prohibitive. In this simple example however, it is possible to retrieve such a path (see [4]). Note that, in [20], we were not able to treat this example (but only a simplified version) due to memory space saturation.

Example 12. This example is a modelization of a part of communication protocol PNCSA (Standard Protocol for Connection to the Authorization System [22]), borrowed from [19]. The Petri net has 38 transitions and 31 counters. The initial marking $\overline{x}_{init}$ has two components equal to 1, and all the other ones null. The deadlock formula is of the form: $\varphi_{dead_1} \vee \cdots \vee \varphi_{dead_4}$.

Previous works. In [19] the system was analyzed through the *minimal coverability graph*, an abstract form of the reachability set, which gives some information, e.g. about boundedness, but cannot certify the absence of deadlocks. We also tried to generate the reachability set using the Presburger based procedure of [20,29], but the system ran out of memory.

New results. The extended Petri net R' is obtained by decomposing R up to level 20. The iterated computation of $Upre_{R'}$ applied to φ_{dead_1} terminates after 53 steps, and takes 15 minutes. $Upre^{*}_{R'}$ contains 376 constraints. Besides, for all $j \leq 53$, one can here again check that all the nonnull coefficients of μ in $pre_{R'^{\mu}}(Upre^{j}_{R'})$ appearing in lower bounds on μ are equal to 1. Therefore $Upre^{*}_{R'} = pre^{*}_{R'} = pre^{*}_{R}$. One constraint of $Upre^{*}_{R'}$ has ϕ_{init} as an instance, which proves that the system has a deadlock. It is here also possible to exhibit a path linking $\overline{x}_{init}$ to a deadlock marking (see [4]).

Note that the existence of this deadlock was unknown so far. There are several possible explanations to this deadlock ("local" but non "global" deadlock, flaws in the modelling, ...) but such an issue is beyond the scope of the paper.

5 Timed Nets

In this section, we consider "timed nets", obtained by adding to timed automata a finite set X of places (or counters) [23]. This allows us to combine known results on timed automata with those of §3 on Petri nets. Note that many other models have been proposed to build timed extensions of Petri nets (see e.g. [28]), but we do not consider them here. We first recall the definitions of timed automata, then treat the case of timed automata with counters (timed nets), with a short discussion on the issue of meta-transitions.

5.1 Timed automata

We consider here Alur-Dill's timed automata [2,1]. As finite state automata, timed automata have a finite set of locations. In addition, they use a finite set Y of variables $y_1, \ldots, y_n$, called *clocks*, which evolve continuously at the same rate. A transition is enabled only if some relation, called *guard*, is satisfied by the current values of the clocks. When a transition is fired, some clocks are reset to fixed values. Moreover, inside each location, the clock values are required to satisfy a relation called *invariant*. Guards and invariants are conjunctions of constraints of the form $y_i \ll a_i$ or $a_i \ll y_i$ with $\ll \in \{<, \leq\}$. Formally, a timed automaton T is a tuple $\langle L, \ell_{init}, Y, I, E\rangle$, where

- L is a finite set of locations,
- $\ell_{init} \in L$ is the *initial location*,
- $Y = \{y_1, \cdots, y_n\}$ is a finite set of clocks ,
- I is a mapping that labels each location ℓ in L with some location invariant, simply written I_ℓ instead of $I(\ell)$ in the following,
- E is a set of *action transitions* of the form e: $\langle(\ell, \ell'), \psi, \rho\rangle$ where ℓ and ℓ' belong to L, ψ is a guard and ρ is a mapping over Y such that, for all $y \in Y$, either $\rho(y) = y$ or $\rho(y) = 0$.

A *state* is a mapping from $\{s\} \times Y$ to $L \times \mathbb{R}^+$ where s denotes a location variable. A set of states will be represented by a formula $\phi(s, \overline{y})$ where $\overline{y}$ is vector $y_1 \cdots y_n$ interpreted as (nonnegative) real-valued variables. The semantics of timed automata are defined by initial formula and reachability relations:
- the *initial formula* $\phi_{init}(s, \overline{y})$ is: $s = \ell_{init} \wedge \overline{y} = 0$.
- the *action reachability relation* via $e : \langle(\ell, \ell'), \psi, \rho\rangle$, written $\xrightarrow{e}$, is such that:
$$(s, \overline{y}) \xrightarrow{e} (s', \overline{y}') \quad \text{iff} \quad s = \ell \wedge s' = \ell' \wedge\ I_s(\overline{y}) \wedge I_{s'}(\overline{y}') \wedge\ \psi(\overline{y})\ \wedge \overline{y}' = \rho(\overline{y}).$$
- the *delay reachability relation*, written $\xrightarrow{\delta}$, is such that:
$$(s, \overline{y}) \xrightarrow{\delta} (s, \overline{y}') \quad \text{iff} \quad \exists \varepsilon \in \mathbb{R}^+\ \ (I_s(\overline{y}) \wedge I_s(\overline{y}') \wedge \overline{y}' = \overline{y} + \varepsilon).$$
- the *reachability relation via automaton T*, written $\xrightarrow{T}$, is $\bigcup_{e \in E}(\xrightarrow{\delta}\ .\ \xrightarrow{e}\ .\ \xrightarrow{\delta})$

We also recall the notion of "clock region" [1].

Definition 13. A formula $\phi(\overline{y})$ is a *clock region* iff it is a conjunction of inequalities of the form: $\quad y_i + d \ll y_j, \quad y_i \ll y_j + d, \quad d \ll y_i, \quad y_i \ll d$
with $y_i, y_j \in Y$, $d \in \mathbb{N}$ and $\ll \in \{<, \leq\}$.

5.2 Timed automata with counters

We now show that proposition 3 also holds for timed automata with counters. For the sake of conciseness we give only results concerning the set *pre* of predecessors, but similar results also hold for *post*. An (extended) action transition f does not go any longer from s to s' over L, but from a pair $(\overline{x}, s)$ to a pair $(\overline{x}', s')$ over $\mathbb{N}^m \times L$ such that: $\overline{x}' = \overline{x} + \overline{c} \wedge \overline{x} \geq \overline{a}$, for a vector $\overline{c}$ of integers and a vector $\overline{a}$ of natural numbers associated with f. A *state* is now a mapping from $X \times \{s\} \times Y$ to $\mathbb{N} \times L \times \mathbb{R}^+$ and a set of states will be represented by a formula $\Phi(\overline{x}, s, \overline{y})$ where $\overline{x}$ is interpreted over $\mathbb{N}$ and $\overline{y}$ over $\mathbb{R}^+$.
A *timed net* U is a tuple $\langle X, \varphi_{init}, L, \ell_{init}, Y, I, F\rangle$, where X is the set of counters, φ_{init} the initial formula for counters, L the set of locations, ℓ_{init} the initial location, I the invariant mapping, Y the set of clocks and F the set of transitions. We define:
- the *initial formula* $\Phi_{init}(\overline{x}, s, \overline{y})$: $\varphi_{init}(\overline{x}) \wedge s = \ell_{init} \wedge \overline{y} = 0$
- the *extended action reachability relation* via f: $\langle(\overline{c}, \overline{a}), (\ell, \ell'), \psi, \rho\rangle$, written $\xrightarrow{f}$:
  $(\overline{x}, s, \overline{y}) \xrightarrow{f} (\overline{x}', s', \overline{y}')$ iff
$\overline{x}' = \overline{x} + \overline{c} \wedge \overline{x} \geq \overline{a} \quad \wedge \quad s = \ell \wedge s' = \ell' \wedge I_s(\overline{y}) \wedge I_{s'}(\overline{y}') \wedge \psi(\overline{y}) \wedge \overline{y}' = \rho(\overline{y})$
- the *delay reachability relation* $\xrightarrow{\delta}$:
  $(\overline{x}, s, \overline{y}) \xrightarrow{\delta} (\overline{x}, s, \overline{y}')$ iff $\exists \varepsilon \in \mathbb{R}^+ \ (I_s(\overline{y}) \wedge I_s(\overline{y}') \wedge \overline{y}' = \overline{y} + \varepsilon)$.
- the *reachability relation via* U, written $\xrightarrow{U}$: $\bigcup_{f \in F}(\xrightarrow{\delta} \ . \ \xrightarrow{f} \ . \ \xrightarrow{\delta})$.

Given a formula Φ and a timed net U, we define $pre_U(\Phi)$ as the result of eliminating s' over L, $\overline{x}'$ over $\mathbb{N}$ and $\overline{y}'$ over $\mathbb{R}$ from
$$\exists \overline{x}', s', \overline{y}' \quad ((\overline{x}, s, \overline{y}) \xrightarrow{U} (\overline{x}', s', \overline{y}') \quad \wedge \quad \Phi(\overline{x}', s', \overline{y}')).$$

We now extend the notions of counter regions and counter class $\mathbf{C}$ as follows:

Definition 14. A formula $\Phi(\overline{x}, \overline{y})$ is a *mixed region* iff it is a conjunction of inequalities of the form:
$$x_i + d \geq x_j, \quad x_i \geq x_j + d, \quad x_i \geq d, \quad d \geq x_i,$$
$$y_i + d \ll y_j, \quad y_i \ll y_j + d, \quad d \ll y_i, \quad y_i \ll d$$
with $x_i, x_j \in X \cup P$, $y_i, y_j \in Y$, $d \in \mathbb{N}$ and $\ll \in \{<, \leq\}$.

Definition 15. A formula $\Phi(\overline{x}, s, \overline{y})$ *belongs to mixed class* $\mathbf{D}$ iff it is a disjunction of the form $\bigvee_{k \in K} \ s = \ell_k \wedge \Phi_k(\overline{x}, \overline{y})$, where K denotes a finite set of indices, ℓ_k is in L, and $\Phi_k(\overline{x}, \overline{y})$ is a mixed region.

Proposition 16. *Let Φ be a formula of $\mathbf{D}$ and U a timed net. Then:*

1. *Variable elimination with Fourier-Motzkin is exact for computing $pre_U(\Phi)$.*
2. *$pre_U(\Phi)$ belongs to $\mathbf{D}$.*
3. *When it terminates, the iterative computation of $pre_U^j(\Phi)$ (using Fourier-Motzkin) yields a formula $pre_U^*(\Phi)$ belonging to $\mathbf{D}$.*

Proof. (sketch) By proposition 3, statements 1 and 2 hold in the special case of an ordinary Petri net ($Y = \emptyset$). It easy to see that they hold also in the special case of an ordinary timed automata ($X = \emptyset$), using the closure property of "clock regions" [1].[4] The truth of statements 1 and 2 for a general timed net U, follows from the combination of these special cases, using the fact that, when computing pre_U, there is no interaction between variables of X and variables of Y in Φ. Statement 3 then follows from statements 1 and 2.

The proposition above allows us to consider all the variables as if they were real ones (including variables of X) and to use Fourier-Motzkin elimination for computing predecessors in class $\mathbf{D}$.

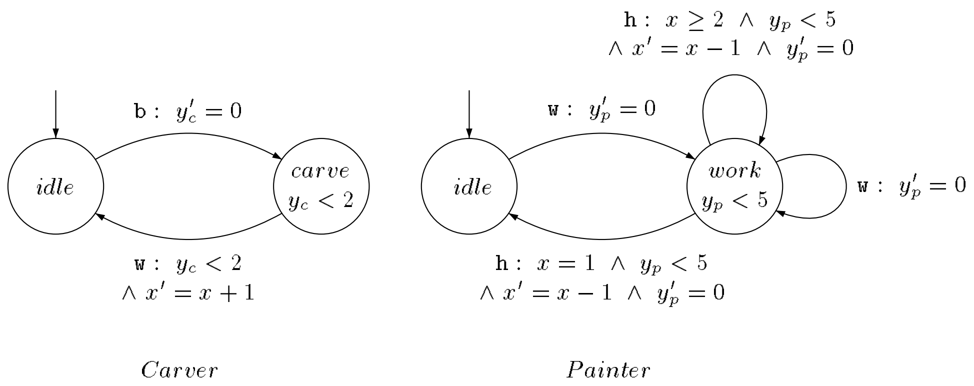

Fig. 3. Horse Manufactory

Example 17. This example, taken from [23], is a simple model of a "Horse Manufactory", where wooden black horses are produced by a carver and a painter. The system is represented in figure 3 by two timed nets, for the carver and the painter respectively. The carver has two locations *idle*, *carve* and one clock y_c, while the painter has two locations *idle* and *work* and the clock y_p. The processes communicate through the transition labeled `w`, which corresponds to the production of a wooden horse by the carver and its reception by the painter. This transition operates on the place x, which counts the number of carved horses that are not yet painted. The carver begins his work upon arrival of a block of wood, corresponding to the transition from *idle* to *carve*, with label `b`, which resets the carver clock y_c. From this point on, carving must be finished in at most 2 time units: invariant $y_c < 2$ is associated with the location *carve* and the transition labeled `w` increments the counter x and returns to location *idle*. With the same transition `w`, the painter first stores the piece. Then, transition `h`

[4] Statement 1 is trivial in this case because all the variables belong to Y, and are real.

represents the production by the painter of one black horse, which decrements x and must occur in at most 5 time units, unless a new wooden horse is produced. Thus, both transitions `w` and `h` reset the painter clock y_p.

We consider properties relative to the maximal delay between arrival of a block of wood and production of a painted horse. The strong property (S) simply requires that this delay be equal to 7: each time an action `b` occurs, there is an action `h` within a delay strictly less than 7 time units. Property (S) is expressed by an "observer automaton" and proved *false* (in the system composed of *Carver*, *Painter* and *Observer*) in seven iterations. The weak version (W) of this property expresses as follows: if `b` occurs and is not followed by another action `b` for 7 time units, then an action `h` occurs within this delay. Property (W) is proved *true* after three iterations (see [4] for details).

5.3 Meta-transitions

The notions of fused and meta-transitions transitions, as defined above in the context of ordinary Petri nets, do not easily extend to timed nets. The problems already arise with timed automata (without counters). One difficulty stems from the fact that the application of two action transitions e_1 and e_2 interleaved with delay-transitions $(\stackrel{\delta}{\rightarrow} . \xrightarrow{e_1} . \stackrel{\delta}{\rightarrow} . \xrightarrow{e_2} . \stackrel{\delta}{\rightarrow})$ cannot be reduced to the application of some "fused" action-transition, say e_3, interleaved with delay-transitions $(\stackrel{\delta}{\rightarrow} . \xrightarrow{e_3} . \stackrel{\delta}{\rightarrow})$, so the situation is radically different from the one prevailing in the case of ordinary Petri nets.[5] Still the finite application of a sequence of action transitions between two states $\overline{y}$ and $\overline{y}'$ defines a linear arithmetic relation, and the transitive closure of such a relation (meta-transition) is still definable in the additive theory of $\mathbb{R}$, as shown by Comon and Jurski ([12], p.274; see also [13]). However the complexity of such a relation is untractable (double exponential in the number of clocks in the worst case). A possible manner to overcome this problem might be to approximate this transitive closure relation using, e.g., methods proposed in [27], but this would deserve further study.

6 Final Remarks

We have given a theoretical justification for using tools designed for the analysis of automata with real-valued variables in order to analyse automata with integer-valued variables such as Petri nets. The constraint solving procedure was very easy to implement in HyTech, as well as the encoding of meta-transitions. Such a task could also have been achieved with a constraint programming language such as CLP($\mathbb{R}$) (as done in [17] in the context of EFSM). Other real-time analysis tools like Uppaal [3] might also be used when no parameter is involved (although encoding meta-transitions may be a problem). To our knowledge, the

[5] In [34] there is an attempt to extend fusion rules à la Berthelot [5] within another model of time Petri nets [6], but with severe conditions for rule application.

idea of using variable elimination over $\mathbb{R}$ instead of $\mathbb{N}$ for computing the set *post** of reachable markings for Petri nets, had not been exploited so far.[6] Experiences show an impressive gain of performance: all the examples from [29] have been reproduced with a speed-up from 10 to 50 times, a deadlock has been found for the first time in the PNCSA protocol (as modeled in [19]), ... The use of real arithmetic with meta-transitions may yield an upper approximation of the computed reachability set. Such an approximation is the price to pay for improving the convergence of *post**. Using real arithmetic can be seen here as an alternative to classical approximation methods as those used in Abstract Interpretation (e.g., "widening" or "convex hull") [14,15,24]. Finally our method extends to Petri nets with inhibitors and to timed nets. However difficulties arise when dealing with fused and meta-transitions for timed nets, and applicability of the approach in this context should be further studied on significant examples.

References

1. R. Alur, C. Courcoubetis and D. Dill. "Model-Checking in Dense Real-Time". *Information and Computation 104:1*, 1993, pp. 2–34.
2. R. Alur and D. Dill. "Automata for Modeling Real-Time Systems". *ICALP'90*, LNCS 443, Springer-Verlag, 1990, pp. 322–335.
3. J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson and W. Yi. "UPPAAL – a Tool Suite for Automatic Verification of Real-Time Systems". *Hybrid Systems III*, LNCS 1066, Springer-Verlag, 1996, pp. 232-243.
4. B. Bérard and L. Fribourg. "Reachability Analysis of (Timed) Petri Nets Using Real Arithmetic". *Technical Report LSV-99-3*, CNRS & Ecole Normale Supérieure de Cachan, March 1999 (http://www.lsv.ens-cachan.fr/Publis/).
5. G. Berthelot. "Transformations and Decompositions of Nets". *Advances in Petri Nets*, LNCS 254, Springer-Verlag, 1986, pp. 359–376.
6. B. Berthomieu, M. Diaz. "Modeling and verification of time dependent systems using time Petri nets". *IEEE Trans. Softw. Eng.* 17:3, 1991, pp. 259–273.
7. B. Boigelot and P. Wolper. "Symbolic Verification with Periodic Sets". *CAV'94*, LNCS 818, Springer-Verlag, 1994, pp. 55–67.
8. G.W. Brams. *Réseaux de Petri: Théorie et Pratique.* Masson, Paris, 1983.
9. T. Bultan, R. Gerber and W. Pugh. "Symbolic Model Checking of Infinite State Systems Using Presburger Arithmetic". *CAV'97*, LNCS 1254, 1997, pp. 400–411.
10. F. Chu and X. Xie. "Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming". To appear in *IEEE Trans. on Robotics and Automation.*
11. 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". *Advances in Petri Nets*, LNCS 483, Springer-Verlag, 1990, pp.79–112.
12. H. Comon and Y. Jurski. "Multiple Counters Automata, Safety Analysis and Presburger Arithmetic". *CAV'98*, LNCS 1427, Springer-Verlag, 1998, pp. 268–279.
13. H. Comon and Y. Jurski. "Timed automata and the theory of real numbers". *CAV'99*, LNCS, Springer-Verlag, 1999. (This volume)

[6] In [11], Fourier-Motzkin elimination is used for computing "p-semiflows" of Petri nets, but p-semiflows are only rough approximations of *post**.

14. P. Cousot and R. Cousot. "Abstract Interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints". *POPL'77*, ACM Press, 1977, pp. 238–252.
15. P. Cousot and N. Halbwachs. "Automatic Discovery of Linear Restraints Among Variables of A Program". *POPL'78*, ACM Press, 1978, pp. 84–97.
16. R. David and H. Alla. *Du Grafcet aux Réseaux de Petri*, Hermès, Paris, 1989.
17. Delzanno and A. Podelski. "Model-Checking in CLP". *TACAS'99*, LNCS 1579, Springer-Verlag, 1999, pp. 223–239.
18. J. Esparza and M. Nielsen. "Decidability Issues for Petri Nets – a Survey". *EATCS Bulletin* 52, 1994.
19. A. Finkel. "The Minimal Coverability Graph for Petri Nets". *Advances in Petri Nets.* LNCS 674, Springer-Verlag, 1993, pp. 211–243.
20. L. Fribourg and H. Olsén. "Proving safety Properties of Infinite State Systems by Compilation into Presburger Arithmetic". *CONCUR'97*, LNCS 1243, Springer-Verlag, 1997, pp. 213–227.
21. L. Fribourg and H. Olsén. "A Decompositional Approach for Computing Least Fixed-Points of Datalog Programs with $\mathbb{Z}$-counters". *Constraints: An International Journal*, 2, 1997, pp. 305–335.
22. GIE CB. "Protocole normalisé de connexion au système d'autorisation: spécifications PNCSA Version 2". Document C *Communication OSI*, 1988.
23. R. Gorrieri and G. Silipandri. "Real-Time System Verification using P/T Nets". *CAV'94*, LNCS 818, Springer-Verlag, 1994, pp. 14–26.
24. N. Halbwachs. "Delay Analysis in Synchronous Programs". *CAV'93*, LNCS 697, Springer-Verlag, 1993, pp.333–346.
25. W. Harvey, P.J. Stuckey and A. Borning. "Compiling Constraint Solving Using Projection". *CP'97*, LNCS 1330, Springer-Verlag, 1997, pp. 491–505.
26. T. Henzinger, P.-H. Ho and H. Wong-Toi. "A user Guide to HyTech". *TACAS'95*, LNCS 1019, Springer-Verlag, 1995, pp. 41–71.
27. W. Kelly, W. Pugh, E. Rosser and T. Shpeisman. *Transitive closure of infinite graphs and its applications.* Technical Report CS-TR-3457, UMIACS-TR-95-48, University of Maryland, 1994.
28. P. Merlin and D.J. Farber. "Recoverability of communication protocols". *IEEE Trans. on Communications* 24:9, Sept. 1976.
29. H. Olsén. *Automatic Verification of Petri Nets in a CLP framework.* Ph.D Thesis, University of Linköping, Sweden, 1997.
30. W. Pugh. "A practical integer algorithm for exact array dependence analysis". *C.ACM* 35:8, 1992, pp. 102–114.
31. A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, 1986.
32. T.R. Shiple, J.H. Kukula and R.K. Ranjan. "A Comparison of Presburger Engines for EFSM Reachability". *CAV'98*, LNCS 1427, Springer-Verlag, 1998, pp. 280–292.
33. R. Shostack. "Deciding linear inequalities by computing loop residues". *J. ACM* 28:4, 1981, pp. 769–779.
34. R. Sloan and U. Buy. "Reduction rules for time Petri nets". *Acta Informatica* 33, 1996, pp. 687–706.
35. P. Wolper and B. Boigelot. "Verifying Systems with Infinite but Regular State Spaces". *CAV'98*, LNCS 1427, Springer-Verlag, 1998, pp. 88–97.
36. M.C. Zhou, F. Dicesare and A.A. Desrochers. "A Hybrid Methodology for Synthesis of Petri Net Models for Manufacturing Systems". *IEEE Trans. on Robotics and Automation* 8:3, 1993, pp. 350–361.