Efficient reachability graph representation of Petri nets with unbounded counters

Franck Pommereau, Raymond Devillers, Hanna Klaudel

Efficient reachability graph representation of Petri nets with unbounded counters

Franck Pommereau, Raymond Devillers, Hanna Klaudel

Efficient reachability graph representation of Petri nets with unbounded counters

Franck Pommereau[1]

LACL, Université Paris 12. 61 avenue du général de Gaulle. 94010 Créteil, France

Raymond Devillers[2]

Département d'Informatique, Université Libre de Bruxelles. CP212, B-1050 Bruxelles, Belgium

Hanna Klaudel[3]

IBISC, FRE 2873 CNRS, Université Evry-Val d'Essonne. 91000 Evry, France

Abstract

In this paper, we define a class of Petri nets, called *Petri nets with counters*, that can be seen as place/transition Petri nets enriched with a vector of integer variables on which linear operations may be applied. Their semantics usually leads to huge or infinite reachability graphs. Then, a more compact representation for this semantics is defined as a symbolic state graph whose nodes possibly encode infinitely many values for the variables. Both representations are shown behaviourally equivalent.

Keywords: Petri nets, unbounded integers, reachability, traces.

1 Introduction

Handling (and in particular model checking) systems with huge or infinite state spaces is known to be usually problematic, especially when one has to manage data with infinitely many values. The problem may be solved for instance when considering effective well-structured systems (where the evolutions are monotonic with respect to some well chosen well quasi order), when the state sets under consideration are upward or downward closed, and thus may be characterised by a finite set of generators [3]. The reachability set is usually not closed itself, but when the data space is regular enough, which is usually the case in practice (even if in all generality reachability is often undecidable), it may happen that it can be represented

[1] Email: franck.pommereau@univ-paris12.fr
[2] Email: rdevil@ulb.ac.be
[3] Email: klaudel@ibisc.univ-evry.fr

using finite structures. This implies to be able to represent finitely infinite sets of states and to compute a possibly infinite set of reachable states in a finite amount of time. For instance, the approach developed in [1] (and implemented in [2]) allows to define data structures to represent sets of integer vectors that can be defined in the Presburger arithmetic and to apply to them linear transformations. It also allows to compute in a finite amount of time the result of an infinite sequence of transformations, which is called in [1] a *meta-transition*. If enough meta-transitions can be executed, the complete state space of a system may possibly be computed (and represented) in a finite amount of time. Of course, not all state spaces can be given a finite representation, nor all meta-transitions can be executed using a finite computation. Indeed, there exist transformations for which this "condensation" is not possible, but as explained in [1], it may be preferable to have a partial solution to a general interesting problem (especially if in practice it usually works) rather than a complete solution for a useless class of problems.

The aim of this paper is to apply these techniques in the context of Petri nets [6]. To do so, we define a class of Petri nets, called *Petri nets with counters* (*PNZ*). They are place/transition (P/T) Petri nets enriched with data in a global vector of integer variables, manipulated through linear transformations. The places, arcs and tokens in PNZ are like in P/T nets, while each transition carry an annotation encoding the linear transformation to perform at each firing. A PNZ can be understood as coding the control state in its places and the data state in its variables. It may also be seen as a special case of coloured Petri nets, that implement the integer variables as integer marked places. Its semantics usually leads to produce infinite reachability graphs.

A more compact state graph construction is defined next. This is made possible by detecting executions that change the data state but not the control one, and executing them as a meta-transition. The result is a compact graph in which each state possibly encodes a huge or infinite number of reachable markings.

The main result of the paper is to show that the compressed state graph preserves the semantics of the system in that it encodes exactly the same set of traces (firing sequences) and the same set of reachable markings.

The basic motivation for specifically introducing a model of Petri nets with unbounded counters, equipped with a compressed semantics, is the aim of efficiently verifying systems modelled using the *causal time* approach where the passing of time is explicitly represented by counting the occurrences of a tick transition [5].

2 Basic notations and definitions

Let $\mathbb{Z}$ denote the set of integers and $\mathbb{D} = \{\bullet\} \uplus \mathbb{Z}$ the set of *data values*, where "$\bullet$" is the Petri net black token. If f and g are two functions on a domain H, we denote by $f \circ g$ their composition, defined for all $a \in H$ as $(f \circ g)(a) \stackrel{\text{df}}{=} f(g(a))$.

Let $n \geq 0$ be an integer and let $\mathbb{V} \stackrel{\text{df}}{=} \{x_0, \ldots, x_{n-1}\}$ be a finite set of *variables* such that $\mathbb{V} \cap \mathbb{D} = \emptyset$. We shall also assume to have chosen an ordering on those variables and denote by $X \stackrel{\text{df}}{=} [x_0, \ldots, x_{n-1}] \in \mathbb{V}^n$ the corresponding vector enumerating all those variables. In the examples we will use either $X = [x, y, z]$ or simply $X = [x]$.

A *linear transformation* L is a pair (C, U) where:

- C is a *linear condition*, *i.e.*, an expression of the form $PX \leq Q$, where P is an $m \times n$ integer matrix, and Q is a vector in $\mathbb{Z}^m$, for $m \geq 0$;
- U is a *linear update*, *i.e.*, an expression of the form $AX + B$ where A is an $n \times n$ integer matrix and $B \in \mathbb{Z}^n$.

Linear conditions allow to express a large variety of (conjunctive) conditions. For instance, $(x < y) \wedge (x + 2y = 3z) \wedge (z > 5)$ is equivalent to $(x - y \leq -1) \wedge (x + 2y - 3z \leq 0) \wedge (-x - 2y + 3z \leq 0) \wedge (-z \leq -6)$, that is linear. Disjunctive conditions, of the kind $\gamma_1 \vee \gamma_2$, and in particular of the kind $a \neq b$ (which is in fact $(a - b \leq -1) \vee (b - a \leq -1)$), are not expressible, but they may be replaced by several linear transformations associating each distinct disjunct to the same update. Linear updates allow to simultaneously assign all the variables in X with linear combinations of their previous values (multiplying variables is thus forbidden). For instance, we may have $(x := 2x + y + 3; y := z; z := y + 1)$. Together with the previously given linear condition, we get the following linear transformation:

$$\left(\begin{bmatrix} 1 & -1 & 0 \\ 1 & 2 & -3 \\ -1 & -2 & 3 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} -1 \\ 0 \\ 0 \\ -6 \end{bmatrix} \quad , \quad \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} \right)$$

In the following, we shall use both matrices or developed formulas.

For a vector V of size n, we denote by $V[x_i]$ its i-th component. In the example above, the linear update U is such that $U[x] = 2x+y+3$, $U[y] = z$ and $U[z] = y+1$.

Let $Z \subseteq \mathbb{Z}^n$ and L be a linear transformation $(PX \leq Q, AX + B)$. The *application* of L on Z, denoted by $L(Z)$, is the subset of $\mathbb{Z}^n$ defined by $\{AY + B \mid Y \in Z, PY \leq Q\}$. Moreover, the *iterated application* of L on Z, denoted by $L^*(Z)$, is the limit (union) of the increasing series of sets $(Z_i)_{i \geq 0}$ where $Z_0 \stackrel{\text{df}}{=} Z$ and $Z_{i+1} \stackrel{\text{df}}{=} Z_i \cup L(Z_i)$.

2.1 Coloured Petri nets

A *Coloured Petri net* (*CPN*) [4] on $\mathbb{D}$ and $\mathbb{V}$ is a tuple $N \stackrel{\text{df}}{=} (S, T, \ell)$ where S is a finite set of *places*, T is a finite set of *transitions* such that $S \cap T = \emptyset$, and ℓ is the *labelling* function on places, transitions and *arcs* (elements from $(S \times T) \cup (T \times S)$):

- for each place $s \in S$, $\ell(s) \subseteq \mathbb{D}$ is the *type* of s;
- for each transition $t \in T$, $\ell(t)$ is an evaluable Boolean expression on $\mathbb{V}$, called the *guard* of t (a condition for its execution);
- for each input arc $(s, t) \in S \times T$, $\ell(s, t)$ is a finite multiset over $\mathbb{D} \cup \mathbb{V}$, representing the values consumed in s when t is executed;
- for each output arc $(t, s) \in T \times S$, $\ell(t, s)$ is a finite multiset of values in $\mathbb{D}$, variables in $\mathbb{V}$ or evaluable expressions on them, representing the values produced in s when t is executed. The difference with the input case is that a computation of new values is allowed by using expressions.

A state of a CPN $N = (S, T, \ell)$ is represented by a *marking* M that is a mapping that associates to each place $s \in S$ a multiset $M(s)$ over $\ell(s)$, that represents the tokens held by each place. A transition $t \in T$ is *enabled* at M, if there exists a *binding* $\rho : \mathbb{V} \to \mathbb{D}$ such that:

- the evaluation of the guard is true: $\mathsf{eval}_\rho(\ell(t)) = \top$;
- for each arc (s,t), there are enough tokens in the place s: $M(s) \geq \mathsf{eval}_\rho(\ell(s,t))$;
- for each arc (t,s), the type of s is respected: $\mathsf{eval}_\rho(\ell(t,s))$ is a finite multiset over $\ell(s)$.

If t is enabled at M for a binding ρ, it may *fire*, leading to a new marking M' defined for each $s \in S$ by $M'(s) \stackrel{\text{df}}{=} M(s) - \mathsf{eval}_\rho(\ell(s,t)) + \mathsf{eval}_\rho(\ell(t,s))$. M' is then called *reachable* in one step from M (notice that, in all generality, many different bindings may lead from M to M' through t). A *trace*,[4] or firing sequence, of a length $k \geq 0$ of a marked CPN (N, M_0) is defined as $M_0 \xrightarrow{t_1,\rho_1} \cdots \xrightarrow{t_k,\rho_k} M_k$, where for all $1 \leq i \leq k$, M_i is reachable in one step from M_{i-1} by firing t_i with the binding ρ_i. M_k is then said to be reachable from M_0 (in k steps); $\mathsf{trace}_k(N, M_0)$ denotes the set of all the traces of length k of (N, M_0) and $\mathsf{trace}(N, M_0) \stackrel{\text{df}}{=} \bigcup_{k\geq 0} \mathsf{trace}_k(N, M_0)$ is the set of all the traces of (N, M_0). The set of traces of a family F of marked CPNs is defined as $\mathsf{trace}(F) \stackrel{\text{df}}{=} \bigcup_{(N,M_0)\in F} \mathsf{trace}(N, M_0)$.

The reachability of the markings from the initial marking M_0 in N can be represented as a (possibly infinite) *reachability graph* which is the directed graph whose nodes are all reachable markings and where each arc represents a transition and a binding producing one marking from another. In this context, the reachability graph can be considered as a compact representation of $\mathsf{trace}(N, M_0)$. In particular, it may remain finite (*i.e.*, have a finite number of reachable markings) even if it admits infinite traces.

Coloured Petri nets may be seen as a generalisation of the P/T nets or, the other way round, a P/T net is a restricted coloured Petri net where:

- for each place $s \in S$, $\ell(s) = \{\bullet\}$;
- for each transition $t \in T$, $\ell(t) = \top$ (true guard);
- for each arc $(a,b) \in (S \times T) \cup (T \times S)$, $\ell(a,b)$ is a finite multiset over $\{\bullet\}$.

Because such a net does not use any variable, the bindings for it are not relevant and will be ignored in the following.

As usual, Petri nets will be represented graphically using circles for the places, rectangles for the transitions and directed links for the arcs. Arcs labelled by empty multisets will be omitted, as well as default annotations: $\top$ for a true guard and $\{\bullet\}$ for a place type or an arc label.

3 Petri nets with Counters

A *Petri net with counters* (*PNZ*) is a P/T net enriched with data represented through a global vector $V \in \mathbb{Z}^n$ of integer values (counters). An additional labelling of the transitions of the net with linear transformations allows to check and update the data at each firing of a transition. More precisely, a PNZ is a tuple $N \stackrel{\text{df}}{=} (S, T, \ell, L)$ where $\lfloor N \rfloor \stackrel{\text{df}}{=} (S, T, \ell)$ is a P/T net and L is mapping associating to each transition t in T a linear transformation $L(t) \stackrel{\text{df}}{=} (C(t), U(t))$. An example PNZ is depicted in the figure 1.

A (aggregated) state of a PNZ N is a pair (M, D) where M is a marking of $\lfloor N \rfloor$ and $D \subseteq \mathbb{Z}^n$, with $D \neq \emptyset$, is a set of possible values for V. A state (M, D) such

[4] Traces should not be confused here with Mazurkiewicz' traces.

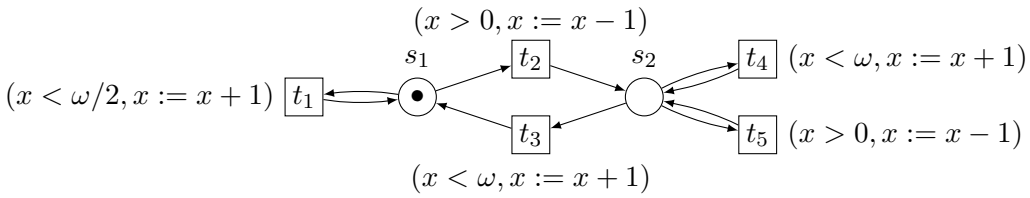

Fig. 1. An example PNZ, for which we assume $X = [x]$; its initial data is $D_0 = \{[0]\}$ and each transition is labelled by its associated linear transformation. The parameter ω is some fixed even non-negative integer.

that D is a singleton is called a *concrete state*; a state in general encodes several concrete states, and possibly infinitely many of them.

3.1 A CPN semantics of PNZs

Let $N = (S, T, \ell, L)$, with $L = (C, U)$ as defined above, be a PNZ and (M, D) be its initial state. It can be translated into a family $\mathsf{cpn}(N, M, D) \stackrel{\text{df}}{=} \{((S', T', \ell'), M_V) \mid V \in D\}$ of marked CPNs which differ only by the initial markings (each such marking corresponding to one concrete initial state), where:

- $S' \stackrel{\text{df}}{=} S \uplus \{s_x \mid x \in \mathbb{V}\}$ and $T' \stackrel{\text{df}}{=} T$;
- for all $s \in S$, $\ell'(s) \stackrel{\text{df}}{=} \{\bullet\}$ and for all $x \in \mathbb{V}$, $\ell'(s_x) \stackrel{\text{df}}{=} \mathbb{Z}$;
- for all $s \in S$, $M_V(s) \stackrel{\text{df}}{=} M(s)$ and for all $x \in \mathbb{V}$, $M_V(s_x) \stackrel{\text{df}}{=} V[x]$;
- for all $t \in T$, $\ell'(t) \stackrel{\text{df}}{=} C(t)$;
- for all $(a, b) \in (S \times T) \cup (T \times S)$, $\ell'(a, b) \stackrel{\text{df}}{=} \ell(a, b)$;
- for all $t \in T$, for all $x \in \mathbb{V}$, $\ell'(s_x, t) \stackrel{\text{df}}{=} \{x\}$ and $\ell'(t, s_x) \stackrel{\text{df}}{=} \{U(t)[x]\}$.

The semantics of a PNZ N in an initial state (M, D) is the set $\mathsf{trace}(\mathsf{cpn}(N, M, D))$ of traces of the corresponding translated CPNs.

An example of such a translation, giving a single CPN, and the corresponding reachability graph are illustrated in the figures 2 and 3, respectively.

In order to compare a concrete state of a PNZ and a marking M of a CPN, we shall denote the latter by a pair $(M_\bullet, V)$, where $M_\bullet$ is M restricted to the places of type $\{\bullet\}$ and $V \in \mathbb{Z}^n$ is defined for each $x \in \mathbb{V}$ by $V[x] \stackrel{\text{df}}{=} M(s_x)$. For example, the marking of the CPN depicted in the figure 2 may be represented as $(\{s_1 \mapsto \{\bullet\}, s_2 \mapsto \emptyset\}, [0])$.

Notice that it could also have been possible to first associate a fresh variable v_s to each place s of N, to associate a value m to v_s whenever $M(s) = m \cdot \{\bullet\}$, to keep a single place with side loops for each transition of N, and to adapt adequately their linear transformations. Indeed, the firing condition of a transition of a P/T net is linear in those new variables, as well as the result of its firing. However, it is generally preferable to keep the original places since the control state of the net has an intuitive meaning and, in many practical systems, the reachable states

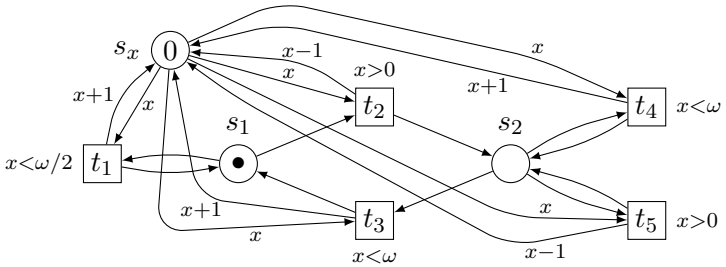

Fig. 2. The CPN translation of the PNZ from the figure 1.

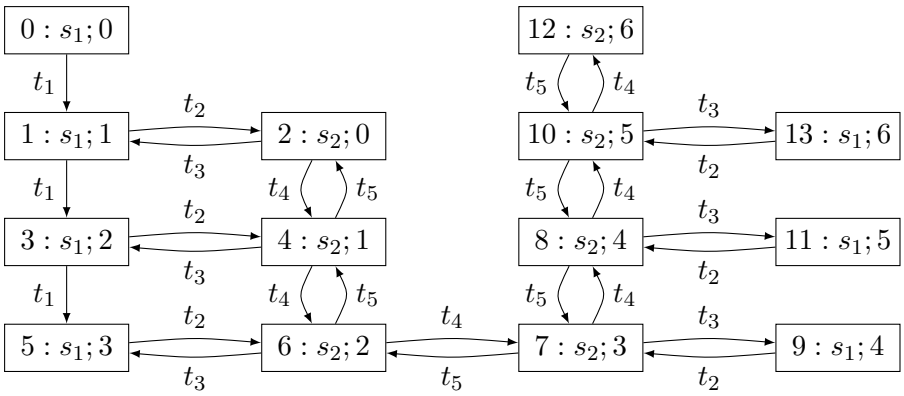

Fig. 3. The reachability graph of the CPN from the figure 2 when $\omega = 6$. Each state is labelled by its number, the place in $\{s_1, s_2\}$ that is marked and the marking of s_x. When ω grows, the left and right parts of the graph, which are connected through the states 6 and 7, become taller while keeping the same structures.

correspond to safe markings of those places (at most one $\bullet$ in each control place).

3.2 A more compact reachability graph

Let $N = (S, T, \ell, L)$ be a PNZ in an initial state (M_0, D_0). Its semantics may be captured in a more compact manner by a symbolic state graph, whose nodes are states of the PNZ possibly aggregating infinitely many markings of the CPNs in $\mathsf{cpn}(N, M_0, D_0)$. More precisely, a transition $t \in T$ is enabled at a state (M, D) if t is enabled in $\lfloor N \rfloor$ at M and at least one vector in D satisfies the condition $C(t)$. The firing of t yields a state (M', D') such that M' is the marking produced by the firing of t in $\lfloor N \rfloor$ and $D' \stackrel{\text{df}}{=} L(t)(D)$. Then, a compacted state graph G may be built as follows:

- (M_0, D_0) is a node of G;
- if (M, D) is a node of G such that (M, D) enables a transition t in N and (M', D') is the state reachable from (M, D) by the firing of t, then, for $D'' \stackrel{\text{df}}{=} L(t)^*(D')$ if $M = M'$, or $D'' \stackrel{\text{df}}{=} D'$ otherwise, we have:
  - · if there is no node (M', D''') in G such that $D'' \subseteq D'''$, then we add the node (M', D'') to G and an arc labelled t from (M, D) to (M', D''),
  - · otherwise, we choose in G an arbitrary node (M', D''') such that $D'' \subseteq D'''$ and we add an arc labelled by t from (M, D) to (M', D''').

Intuitively, if a transition t can fire at (M, D) leading to (M, D'), it means that t can fire (possibly many times) from (M, D'') with $D'' \stackrel{\text{df}}{=} L(t)^*(D')$, at least if C allows it. So, the idea is to aggregate the effect of all the firing sequences obtained by iterating t (there is potentially an infinity of them), considered as a meta-transition, into one symbolic state (M, D''). Because of the application of $L(t)^*$, a loop arc on (M, D'') labelled by t will always be constructed when the successors of (M, D'') will be computed. On the other hand, if from (M, D) a computed state is (M', D''), and a state (M', D'''), with $D'' \subseteq D'''$, is already present in G, then (M', D'') is covered by (M', D''') and does not need to be added. It may also happen that a computed state covers some states already present in G (when $D''' \subset D''$). This point will be discussed in section 5.

It should be noted that this construction does not guarantee that G is unique (nor even finite); it may also depend on the order in which nodes and transitions are

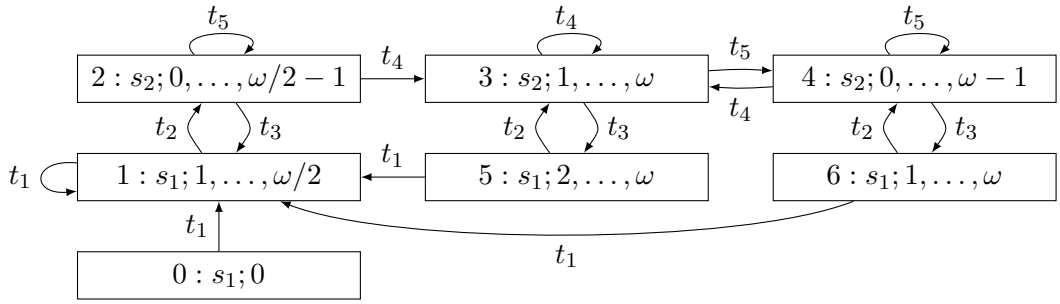

Fig. 4. A compacted state graph of the PNZ from the figure 1 when $6 \leq \omega < \infty$. Each state is labelled by its rank in the construction, the marked place (either s_1 or s_2) and the set of possible values for x. Notice that the node 5 could have pointed to 6 instead of 1, if 6 was constructed before 5.

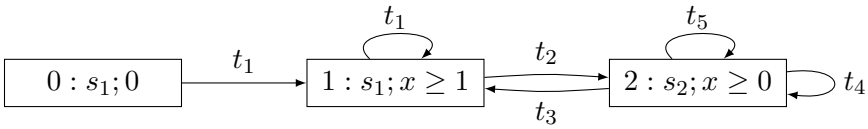

Fig. 5. The state graph of the PNZ from the figure 1 when ω is infinite, *i.e.*, when all the conditions involving ω are dropped.

considered as well as on which D''' is chosen when several are possible. However, this will have no consequence on the desired properties. We denote by $G(N, M_0, D_0)$ any possible graph built as above. Examples of such state graphs are depicted in figures 4 and 5.

As a state in a graph $G \stackrel{\text{df}}{=} G(N, M_0, D_0)$ may aggregate several concrete states, a path starting from (M_0, D_0) may actually encode several *elementary paths*. (We will show later on that these elementary paths actually corresponds to traces for the PNZ.) More precisely, for $k \geq 0$, let $P_k(G)$ be the set of all the paths of length k in G that start at (M_0, D_0). Then, if $\pi \stackrel{\text{df}}{=} (M_0, D_0) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (M_k, D_k)$ is a path in $P_k(G)$ (possibly reduced to the initial state if $k = 0$), then we define its unfolding $\mathsf{unf}(\pi)$ as the set of all the elementary paths of length k represented by π, *i.e.*, sequences of transitions between concrete states that can actually be fired. More precisely, $\mathsf{unf}(\pi)$ comprises all the elementary paths $(M_0, V_0) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (M_k, V_k)$ for all $V_i \in D_i$ $(0 \leq i \leq k)$ and such that $L(t_j)(V_{j-1}) = V_j$ (for $1 \leq j \leq k$). The set of elementary paths of length k is given by $\mathsf{path}_k(G) \stackrel{\text{df}}{=} \bigcup_{\pi \in P_k(G)} \mathsf{unf}(\pi)$. The set of zero-length elementary paths is thus $\mathsf{path}_0(G) = \{(M_0, V) \mid V \in D_0\}$. Finally, the set of elementary paths of G (which we shall also call the PNZ elementary paths) is defined as $\mathsf{path}(G) \stackrel{\text{df}}{=} \bigcup_{k \geq 0} \mathsf{path}_k(G)$.

It should be noted that there may exists paths in the compacted state graph that do not correspond to any elementary path, *i.e.*, that cannot actually be executed. This is the case for instance in the graph of the figure 5 where the path $0 \xrightarrow{t_1} 1 \xrightarrow{t_2} 2 \xrightarrow{t_5} 2$ is unfolded to an empty set (after t_1, t_2 we have $x = 0$ thus t_5 is disabled).

4 Behavioural equivalence

A PNZ elementary path $\pi \stackrel{\text{df}}{=} (M_0, V_0) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (M_k, V_k)$ is *equivalent* to a CPN trace $\pi' \stackrel{\text{df}}{=} (M'_0, V'_0) \xrightarrow{t'_1, \rho_1} \cdots \xrightarrow{t'_k, \rho_k} (M'_k, V'_k)$ of the same length k, denoted $\pi \sim \pi'$, if $M'_i = M_i$, $V'_i = V_i$ and $t_i = t'_i$ for $0 \leq i \leq k$. A set Π of PNZ elementary paths is *equivalent* to a set Π' of CPN traces, denoted $\Pi \sim \Pi'$, if there exists a bijection $\beta : \Pi \to \Pi'$ such that $\pi \sim \beta(\pi)$ for all $\pi \in \Pi$.

Theorem 4.1 *Let N be a PNZ in a state (M, D). Then, for any state graph*

$G \stackrel{\text{df}}{=} G(N, M, D)$, *we have* $\mathsf{path}(G) \sim \mathsf{trace}(\mathsf{cpn}(N, M, D))$.

This property is proved by induction on the length of the elementary paths and of the traces. Let N_Z be a PNZ in the initial state (M_0, D_0) and $G \stackrel{\text{df}}{=} G(N_Z, M_0, D_0)$ an arbitrary state graph of N_Z. Without loss of generality, we assume that D_0 is a singleton $\{V_0\}$. Then, $\mathsf{cpn}(N_Z, M_0, D_0) = \{(N_C, (M_0, V_0))\}$. By definition of cpn, N_Z and N_C have the same sets of transitions.
Basic case: $k = 0$. By definition, we have $\mathsf{path}_0(G) = \{(M_0, V) \mid V \in D_0\} = \{(M_0, V_0)\} = \mathsf{trace}_0(N_C, (M_0, V_0))$.
Induction hypothesis. Assume $\mathsf{path}_i(G) \sim \mathsf{trace}_i(N_C, (M_0, V_0))$, for all i such that $0 \leq i \leq k$.
Induction step: $k+1$. It is worth noting that in a trace $(M_0, V_0) \xrightarrow{t_1,\rho_1} \cdots \xrightarrow{t_k,\rho_k} (M_k, V_k)$ of a CPN, the bindings ρ_i are entirely determined by V_{i-1} for $1 \leq i \leq k$: by definition of cpn, in particular how arcs are set, we must have $\rho_i = \{x \mapsto V_{i-1}[x] \mid x \in \mathbb{V}\}$. For this reason, we shall not consider the bindings in the following.

($\Rightarrow$) Let $\pi \stackrel{\text{df}}{=} (M_0, V_0) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (M_k, V_k) \xrightarrow{t_{k+1}} (M_{k+1}, V_{k+1})$ be an elementary path in $\mathsf{path}_{k+1}(G)$. By induction hypothesis, we have a unique trace $(M_0, V_0) \xrightarrow{t_1,\rho_1} \cdots \xrightarrow{t_k,\rho_k} (M_k, V_k)$ in $\mathsf{trace}_k(N_C, M_0, V_0)$ that is equivalent to the prefix $(M_0, V_0) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (M_k, V_k)$ of π. By definition of cpn, in particular how guards and arcs to the places s_x's are set, since t_{k+1} can fire in N_Z at (M_k, V_k), this is also the case in N_C at (M_k, V_k) and its firing produces the corresponding state (M_{k+1}, V_{k+1}). Thus, $\pi' \stackrel{\text{df}}{=} (M_0, V_0) \xrightarrow{t_1,\rho_1} \cdots \xrightarrow{t_k,\rho_k} (M_k, V_k) \xrightarrow{t_{k+1},\rho_{k+1}} (M_{k+1}, V_{k+1})$ is a trace in $\mathsf{trace}_{k+1}(N_C, M_0, V_0)$, it is unique and $\pi \sim \pi'$.

($\Leftarrow$) This may be exhibited essentially by a symmetric argument. □

4.1 Reachability equivalence

By theorem 4.1, a state graph is allowed to encode more than the actually reachable markings; indeed, because of the unfolding, only the executable elementary paths are kept. We show now that the state graph contains exactly the actually reachable states, each such state being usually reachable by only a subset of the paths in the graph that lead to it.

Corollary 4.2 *Let N be a PNZ in a state (M_0, D_0) and a corresponding compacted state graph G. For all node (M, D) in G and all V in D, there exists a marked net $(N', (M', V')) \in \mathsf{cpn}(N, M_0, D_0)$ such that (M, V) is reachable in N' from (M', V'). Conversely, for all marked net $(N', (M', V')) \in \mathsf{cpn}(N, M_0, D_0)$ and all (M, V) reachable in N' from (M', V'), there exists a node (M', D') in G such that $V' \in D'$.*

($\subseteq$) Holds by construction of G: it creates only states that are actually reached from the initial one, through finite (but possibly iterated) sequences of transitions.

($\supseteq$) By theorem 4.1, equivalent executions lead to equivalent states. □

5 Implementation issues

We have produced a prototype implementation that computes a compacted state graph for a PNZ. The sets of integer vectors are stored using the NDD structure

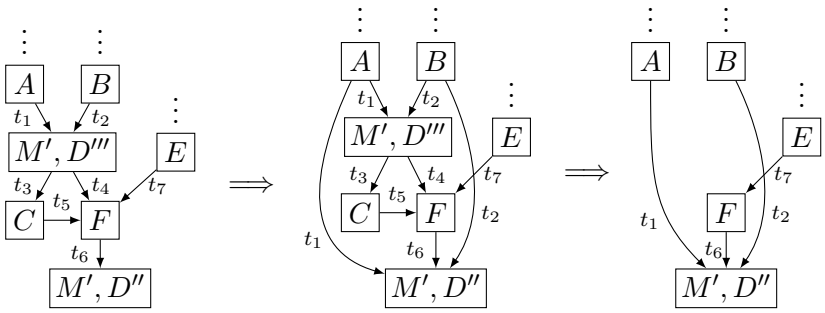

Fig. 6. Left: (M', D'') is added and covers (M', D'''). Middle: A and B are connected to the new state. Right: (M', D''') is removed, then C becomes unreachable and is removed too.

defined in [1] and implemented in the library Lash [2]. In some cases, our implementation ends up with an error, when a transformation cannot be iterated by Lash (as explained in the introduction, this approach is not complete). Moreover, we cannot guarantee that it will terminate as this is a problem that is undecidable in general. However, sufficient conditions for the termination have been given in [1] and it is left as a future work to study how they could be applied in our setting.

We also implemented an optimisation with respect to the definitions presented above. If an existing state is covered by a newly computed one, the former may be replaced by the latter. It implies that the successors states of the replaced one have to be recomputed, as well as their own successors, and so on. Let (M', D'') be the new state and (M', D''') the existing one such that $D''' \subset D''$. For each state (M, D), which is a predecessor of (M', D''') through an arc labelled by t, we create an arc from (M, D) to (M', D'') also labelled by t. Then, (M', D''') is removed, as well as all its successors that become unreachable from the initial node, and so on. This is illustrated in figure 6.

Doing so, we achieve the two expected goals: (M', D''') is replaced by (M', D'') and its successors will be recomputed. Indeed, in the figure 6, since C has been removed, a generalisation of it will be generated when the successors of (M', D'') will be computed. This also holds for F but it could not be removed because it has a predecessor E. If later on we find a generalisation of F as a successor of (M', D''), this new node will replace F, and E will be connected to it as just shown.

We also introduced another optimisation: since (M', D'') generalises (M', D'''), any execution that led from (M', D''') to (M', D'') could be performed again and again: we have discovered a cycle in the state graph. Thus, we can iterate the transformation that created D'' from D'''. In our example, we obtained D'' as $L_1(D''')$ and also as $L_2(D''')$, where[5] $L_1 \stackrel{\text{df}}{=} L(t_6) \circ L(t_4)$ and $L_2 \stackrel{\text{df}}{=} L(t_6) \circ L(t_5) \circ L(t_3)$, since two paths exist from (M', D''') to (M', D''). It is thus possible to execute a meta-transition that corresponds to the iteration of L_1 or L_2 or any combination of them applied to D''. Unfortunately, we cannot use $(L_1 \vee L_2)^*$ which is not an operation available on linear transformations. In our implementation, we compute the composition $L \stackrel{\text{df}}{=} L_1 \circ L_2$ and to consider $L^*(D'')$ instead of D'' when replacing (M', D'''). This is not the only solution but there exist an infinite number of possible combinations of L_1 and L_2, so we have chosen to consider one of them. The properties of this optimisation should be studied in the future.

[5] Notice that, when we chain linear transformations, we not only have to compose their updates, but also their conditions; fortunately, both remain linear.

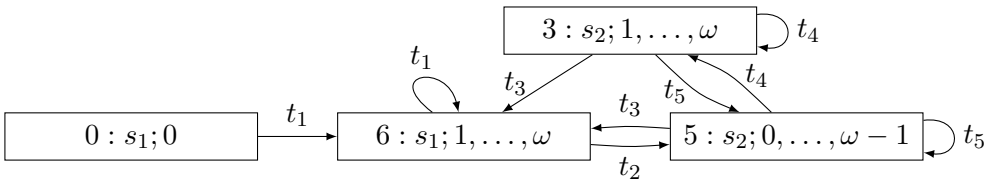

Fig. 7. The state graph after the removing of the covered states: with respect to the graph of the figure 4, the states 1 and 4 have been removed since they are covered by the state 6 and the state 2 has been removed since it is covered by the state 5.

It should be stressed that these optimisations do not change the elementary paths encoded by the graph. Indeed, all the elementary paths based on $A \xrightarrow{t_1} (M', D''')$ are now encoded by $A \xrightarrow{t_1} (M', D'')$. Even if D'' has more values than D''', only the values that can actually be computed from A will be used in the elementary paths. Moreover, it is obvious that the optimisations have no impact on the reachability. As a result of these optimisations, we obtain the graph depicted in the figure 7.

6 Conclusion

We presented here the class of *Petri nets with counters* (*PNZ*) as P/T nets enriched with unbounded (both ends) integer variables on which linear operations may be applied. The semantics of a PNZ is given by that of a family of coloured Petri nets that implement the integer variables as integer marked places. We have presented a compact state graph construction allowing to aggregate many reachable markings (possibly infinitely many) into a single state. This graph has been proved to preserve the trace semantics and to encode exactly the set of reachable markings. Finally, we introduced optimisations allowing to produce even more compact state graphs.

Our next step will be a series of case studies in order to show that, even if not complete, this approach is usable and efficient in practice; in particular for systems with finite but large state spaces. Our first experiments show that it is the case for instance for the nets with causal time [5], even when we consider unbounded counters.

Future works should study the question of the termination of the construction, which is directly related to knowing whether a state graph is finite or not. Moreover, we should study carefully the optimisation introduced in the section 5 that execute a meta-transition when a cycle is discovered in the state graph. The theorem 4.1 being independent on the choice of the considered state graph G, it may also be possible during the construction of G to use adequate conditions that lead to smaller graphs. Another interesting work will be to generalise our approach to the other data structures defined in [1]: subsets of $\mathbb{R}^n$ and unbounded FIFO queues of integers.

References

[1] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces.* PhD Thesis, Univ. of Liège, 1999.

[2] B. Boigelot. *The Liège Automata-based Symbolic Handler.* http://www.montefiore.ulg.ac.be/~boigelot/research/lash/

[3] G. Geeraerts. *Coverability and Expressiveness Properties of Well-structured Transition Systems.* PhD thesis, Université Libre de Bruxelles, 2007.

[4] K. Jensen. *Coloured Petri nets. basic concepts, analysis methods and practical use.* EATCS Monographs on Theoretical Computer Science, 1. Springer, 1992.

[5] F. Pommereau. *Causal time calculus.* FORMATS'03, LNCS 2791, Springer, 2004

[6] W. Reisig. *Petri nets.* EATCS Monographs on Theoretical Computer Science. Springer, 1985.