Comparing Petri Data Nets and Timed Petri Nets

Rémi Bonnet, Alain Finkel, Serge Haddad,
Fernando Rosa-Velardo

December 2010

Research report LSV-10-23

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Comparing Petri Data Nets and Timed Petri Nets

R. Bonnet[1], A. Finkel[1], S. Haddad[1⋆], F. Rosa-Velardo[2⋆⋆]

[1] Ecole Normale Supérieure de Cachan, LSV, CNRS UMR 8643, Cachan, France
E-mail: {remi.bonnet,alain.finkel,serge.haddad}@lsv.ens-cachan.fr

[2] Sistemas Informáticos y Computación, Universidad Complutense de Madrid
E-mail: fernandorosa@sip.ucm.es

Abstract. Well-Structured Transitions Systems (WSTS) constitute a generic class of infinite-state systems for which several properties like coverability remain decidable. The family of coverability languages that they generate is an appropriate criterium for measuring their expressiveness. Here we establish that Petri Data nets (PDNs) and Timed Petri nets (TdPNs), two powerful classes of WSTS are equivalent w.r.t this criterium.

1 Introduction

WSTS. Infinite-state systems appear in a lot of models and applications: stack automata, counter systems, Petri nets or VASSs, reset/transfer Petri nets, fifo (lossy) channel systems, parameterized systems. Among these infinite-state systems, a part of them, called Well-Structured Transition Systems (WSTS) [6] enjoys two nice properties: there is a well quasi-ordering (wqo) on the set of states and the transition relation is monotone with respect to this wqo.

The theory of WSTS has been successfully applied for the verification of safety properties of numerous infinite-state models like Lossy Channel Systems, extensions of Petri Nets like reset/transfer and Affine Well Nets [7], or broadcast protocols. Most of the positive results are based on the decidability of the coverability problem (whether an upward closed set of states is reachable from the initial state) for WSTS, under natural effectiveness hypotheses. The reachability problem, on the contrary, is undecidable even for the class of Petri nets extended with reset or transfer transitions.

Expressiveness. Well Structured Languages were introduced as a measure of the expressiveness of subclasses of WSTS. More precisely, the language of an

⋆ The first three authors are partially supported by the Agence Nationale de la Recherche, AVERISS (grant ANR-06-SETIN-001) and AVERILES (grant ANR-05-RNTL-002).

⋆⋆ The last author is partially supported by the MEC Spanish project DESAFIOS10 TIN2009-14599-C03-01, and Comunidad de Madrid program PROMETIDOS S2009/TIC-1465.

instance of a model is defined as the class of *finite* words accepted by it, with *coverability* as accepting condition, that is, generated by traces that reach a state which is bigger than a given final state. Convincing arguments show that the class of coverability languages is the right one. For instance, though reachability languages are more precise than coverability languages, the class of reachability languages is RE for almost all Petri Nets extensions containing Reset Petri Nets or Transfer Petri Nets.

Contribution. We show here that Petri Data Nets [2] and Timed Petri Nets [3] are equivalent.

Related work. Coverability languages have been used to discriminate the expressive power of several WSTS, like Lossy Channel Systems or several monotonic extensions of Petri Nets. In [8] several pumping lemmas are proved to discriminate between extensions of Petri Nets. In [1, 2] the expressive power of Petri Nets is proved to be strictly below that of Affine Well Nets, and Affine Well Nets are proved to be strictly less expressive than Lossy Channel Systems. Similar results are obtained in [11], though some significant problems are left open, like the distinction between ν-Petri Nets [10] and Data Nets [9].

Outline. The rest of the paper is organized as follows. In section 2 we introduce wqos, WSTS and their languages. Section 3 presents the definition of PDNs and TdPNs. Finally we prove the equivalence of and PDNs and TdPNs in section 4.

2 Preliminaries and WSTS

Well Quasi Orders. $(X, \leq_X)$ is a *quasi-order* (qo) if $\leq_X$ is a reflexive and transitive binary relation on X. A *partial order* (po) is an antisymmetric qo. The *downward closure* of a subset $A \subseteq X$ is defined as $\downarrow A = \{x \in X \mid \exists x' \in A,\ x \leq x'\}$. A subset A is *downward closed* iff $\downarrow A = A$. A qo $(X, \leq_X)$ is a *well quasi-order* (wqo) if for every infinite sequence $x_0, x_1, \ldots \in X$ there are i and j with $i < j$ such that $x_i \leq x_j$. Equivalently, a qo is a wqo when there are no strictly decreasing (for inclusion) sequences of downward closed sets.

We will shorten $(X, \leq_X)$ to X when the underlying order is obvious. Similarly, $\leq$ will be used instead of $\leq_X$ when X can be induced from the context.

If X and Y are wqos, their cartesian product, denoted $X \times Y$ is well ordered by $(x, y) \leq_{X\times Y} (x', y') \iff x \leq_X x' \wedge y \leq_Y y'$. Their disjoint union, denoted $X \uplus Y$ is well ordered by :

$$z \leq_{X\uplus Y} z' \iff \begin{cases} z, z' \in X \\ z \leq_X z' \end{cases} \quad \text{or} \quad \begin{cases} z, z' \in Y \\ z \leq_Y z' \end{cases}$$

A po $(X, \leq)$ is *total* (also called *linear*) if for any $x, x' \in X$ we have either $x \leq x'$ or $x' \leq x$. If $(X_i, \leq_i)$ are total po for $i \in \mathbb{N}$ we can define the (irreflexive) total order $<_{lex}$ in $\bigcup_k X_1 \times \ldots \times X_k$ by $(x_1, \ldots, x_p) <_{lex} (x'_1, \ldots, x'_q)$ iff there is $i \in \{1, \ldots, min(p, q)\}$ such that $x_j = x'_j$ for $j < i$ and $x_i <_i x'_i$ or $(x_1, \ldots, x_p) =$

$(x'_1, \ldots, x'_p)$ and $q > p$. Then $\leq_{lex}$ given by $x \leq_{lex} x'$ iff $x = x'$ or $x <_{lex} x'$ is a total order.

Functions. Given a partial function (shortly: function) $f : X \mapsto Y$, the *domain* of f is defined by $dom(f) = \{x \in X \mid \exists y \in Y,\ f(x) = y\}$ and its *range* by $range(f) = \{y \in B \mid \exists x \in A,\ f(x) = y\}$. A function f is *total* if $dom(f) = A$. Total functions are called *mappings*. A function f is *injective* if for all x, x', $f(x) = f(x') \implies x = x'$ and *surjective* if $range(f) = Y$. Finally, if X and Y are quasi-ordered (shortly: ordered), f is *monotonic* if $x \leq_X y \implies f(x) \leq_Y f(y)$. It is an *order embedding* (shortly: embedding) if $\varphi(x) \leq_Y \varphi(x') \iff x \leq_X x'$. A bijective order embedding is called an *order isomorphism* (shortly: isomorphism).

Multisets. Given an arbitrary set X, we denote by $X^{\oplus}$ the set of finite multisets of X, that is, the set of mappings $m : X \to \mathbb{N}$ with a finite support $sup(m) = \{x \in X \mid m(x) \neq 0\}$. We use the set-like notation $\{|...|\}$ for multisets when convenient, with $\{|x^n|\}$ describing the multiset containing x n times. If X is a wqo then so is $X^{\oplus}$ ordered by $\leq_{\oplus}$ defined by $\{|x_1, \ldots, x_n|\} \leq_{\oplus} \{|x'_1, \ldots, x'_m|\}$ if there is an injection $h : \{1, \ldots, n\} \to \{1, \ldots, m\}$ such that $x_i \leq x'_{h(i)}$ for each $i \in \{1, \ldots, n\}$.

Words. Given an arbitrary set X, any $u = x_1 \cdots x_n$ with $n \geq 0$ and $x_i \in X$, for all i, is a finite word on X. We denote by X^* the set of finite words on X. If $n = 0$ then u is the empty word, which is denoted by ϵ. A language L on X is a subset of X^*. Given L and L' two languages on X^*, we define the language $LL' = \{uv \mid u \in L, v \in L'\}$. If X is a wqo then so is X^* ordered by $\leq_{X^*}$ which is defined as follows: $x_1 \ldots x_n \leq_{X^*} x'_1 \ldots x'_m$ if there is a strictly increasing mapping $h : \{1, \ldots, n\} \to \{1, \ldots, m\}$ such that $x_i \leq x'_{h(i)}$ for each $i \in \{1, \ldots, n\}$.

WSTS. A *Labelled Transition System* is a tuple $\mathcal{S} = \langle X, \Sigma, \to \rangle$ where X is the set of states, Σ is the labelling alphabet and $\to \subseteq X \times (\Sigma \cup \{\epsilon\}) \times X$ is the transition relation.

We write $x \xrightarrow{a} x'$ to say that $(x, a, x') \in \to$. This relation is extended for $u \in \Sigma^*$ by $x \xrightarrow{u} x' \iff x \xrightarrow{a_1} x_1 ... x_{k-1} \xrightarrow{a_k} x'$ and $u = a_1 a_2 \cdots a_k$ (note that some a_i's can be equal to ϵ). A *Well Structured Transition System* (shortly a WSTS) is a tuple $S = (X, \Sigma, \to, \leq)$, where $(X, \Sigma, \to)$ is a labelled transition system, and $\leq$ is a wqo on X, satisfying the following monotonicity condition: for all $x_1, x_2, x'_1 \in X, u \in \Sigma^*$, $x_1 \leq x'_1$, $x_1 \xrightarrow{u} x_2$ implies the existence of $x'_2 \in X$ such that $x'_1 \xrightarrow{u} x'_2$ and $x_2 \leq x'_2$. For a class $\mathbf{X}$ of wqos, we will denote by $WSTS_{\mathbf{X}}$ the class of WSTS with state space in $\mathbf{X}$, or just $WSTS_X$ for $WSTS_{\{X\}}$.

Coverability and Reachability Languages. Coverability (resp. reachability) languages are defined as the set of finite sequences which arrive in a state x such that $x \geq x_f$ (resp. $x = x_f$) for x_f a final state. Formally, given a quasi-ordered labelled transition system $\mathcal{S}$ and two states x_0 and x_f, the coverability language is $L(\mathcal{S}, x_0, x_f) = \{u \in \Sigma^* \mid x_0 \xrightarrow{u} x,\ x \geq x_f\}$ while the reachability language is $L_R(\mathcal{S}, x_0, x_f) = \{u \in \Sigma^* \mid x_0 \xrightarrow{u} x_f\}$. Convincing arguments show that the

class of coverability languages is the right one: though reachability languages are more precise than coverability languages, the class of reachability languages is the set of recursively enumerable languages for almost all Petri nets extensions containing reset Petri nets or transfer Petri nets.

On the other hand, infinite coverability languages are not satisfactory either. A sensible accepting condition in this case could be repeated coverability, that is, the capacity of covering a given marking infinitely often, in the style of Büchi automata. However, analogously to what happens with reachability, repeated coverability is generally undecidable, which makes ω-languages a bad candidate to study the relative expressive power of WSTS. Moreover, infinite coverability languages are less standard for the classification of WSTS. Finally, all trace languages are coverability languages in taking the entire set of states as the final upward closed set.

For two classes of WSTS, $\mathbf{S}_1$ and $\mathbf{S}_2$, we write $\mathbf{S}_1 \preceq \mathbf{S}_2$ whenever for every language L recognized by a system $\mathcal{S}_1 \in \mathbf{S}_1$, there exists another system $\mathcal{S}_2 \in \mathbf{S}_2$ and two states s_i, s_f such that $L(\mathcal{S}_2, s_i, s_f) = L$. When $\mathbf{S}_1 \preceq \mathbf{S}_2$ and $\mathbf{S}_2 \preceq \mathbf{S}_1$, one denotes the equivalence of classes by $\mathbf{S}_1 \simeq \mathbf{S}_2$.

3 Petri Nets extensions

Many extensions of Petri nets in which tokens carry data have been defined in the literature, in order to gain expressive power for better modeling capabilities. Data Nets (*DN*) [9] are a monotonic extension of Petri nets in which tokens are taken from a linearly ordered and dense domain, and transitions can perform whole place operations like transfers, resets or broadcasts. Here we will work with a subclass of *DN*, called *Petri Data Net* (*PDN*), in which whole place operations are not allowed, because $DN \simeq PDN$ [2].

3.1 Petri Data Nets

We denote by $\mathbf{0}$ the null vector in any $\mathbb{N}^k$ and for a word $w = x_1 \cdots x_n$ we write $|w| = n$ and $w(i) = x_i$. Σ is a finite alphabet and Σ_ε denotes the alphabet enlarged with the empty word ε.

Definition 1 (Petri Data Nets). *A PDN is a tuple $N = (P, T, F, H, \lambda)$ where:*

- *P is a finite set of places,*
- *T is a finite set of transitions,*
- *For all $t \in T$, $F_t, H_t \in \left(\mathbb{N}^{|P|}\right)^*$ with $|F_t| = |H_t|$,*
- *$\lambda : T \to \Sigma_\varepsilon$ is a labelling function.*

A marking s of N is a finite sequence of vectors in $\mathbb{N}^{|P|} \setminus \mathbf{0}$. A sequence $s' \in \left(\mathbb{N}^{|P|}\right)^*$ is a $\mathbf{0}$-extension of a marking s (or s is the $\mathbf{0}$-contraction of s') if s can be obtained by removing all the occurrences of $\mathbf{0}$ from s'.

Definition 2 (Firing rule of PDNs). *Let N be a PDN and s, s' be markings of N. We write $s \xrightarrow{t} s'$ for $t \in T$ with $|F_t| = |H_t| = n$ if:*

– s *is the* $\mathbf{0}$*-contraction of* $u_0x_1u_1 \cdots u_{n-1}x_nu_n$
  with $u_i \in (\mathbb{N}^{|P|} \setminus \mathbf{0})^*$ *and* $x_i \in \mathbb{N}^{|P|}$,
– $x_i \geq F_t(i)$ *and* $y_i = (x_i - F_t(i)) + H_t(i)$ *for* $i \in \{1, \ldots, n\}$,
– s' *is the* $\mathbf{0}$*-contraction of* $u_0y_1u_1 \cdots u_{n-1}y_nu_n$

Considering that tokens carry data taken from a linearly ordered and dense domain, a *PDN* can be graphically depicted as a Petri net where arcs are labelled with bags of variables that are totally ordered. For instance, if $P = \{p_1, p_2\}$ and $X < Y$ are the only variables that label arcs adjacent to a transition t, then $|F_t| = |H_t| = 2$, and if $F_t(1) = (n_1, n_2)$ then firing t removes n_1 tokens from p_1 and n_2 tokens from p_2, carrying the datum d_x to which X is instantiated. Analogously, $F_t(2) = (m_1, m_2)$ specifies the number of tokens that carry the datum d_y ($> d_x$) to which Y is instantiated that have to be removed (and similary for H_t). The corresponding labels would be $n_1X + m_1Y$ (resp. $n_2X + m_2Y$) on the arc from p_1 (resp. p_2) to t. Let us remark that we can assume w.l.o.g. that the variables are not totally ordered, since any transition that does not fulfill this requirement could be duplicated in order to ensure it. For instance to simulate a transition with two unordered variables X and Y, we use three transitions: one with $X < Y$, one with $Y > X$ and the last one where Y is substituted by X on the labels of arcs surrounding the transition.

3.2 Timed Petri Nets

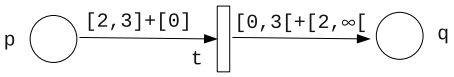

Fig. 1. A timed Petri net

The class of Timed Petri nets (*TdPN*) [3] is a powerful formalism for the modelling of timed systems. In a *TdPN*, tokens have an age. So the arcs of such nets are labelled by bags of intervals as in figure 1. For instance, in order to fire t, one needs to consume two tokens in p, one with age between 2 and 3 and the other with age 0. Once the transition is fired, two tokens are produced in q where the age of one token can (non deterministically) be chosen less than 3 and the other greater or equal than 2. When time elapses, ages of tokens are uniformly increased by this duration.

While *TdPN* is a class of WSTS over an uncountable wqo, the coverability problem is decidable using a symbolic representation of "equivalent" states with representations in a countable wqo. Moreover, when focusing on untimed coverability languages, the class of *TdPN* can be compared to other classes of WSTS. Since the symbolic wqo of *TdPN* is close to the one of *PDN* a natural issue is the comparison of these classes.

We now formally define *TdPN*s. In the sequel, $\mathcal{I}$ is the set of intervals with bounds in $\mathbb{N} \cup \{\infty\}$.

Definition 3. *A* timed Petri net *(TdPN for short) $\mathcal{N}$ is a tuple $(P, T, \mathsf{Pre}, \mathsf{Post}, \lambda)$ where:*

- *P is a finite set of places,*
- *T is a finite set of transitions with $P \cap T = \emptyset$,*
- *Pre, the backward incidence mapping, is a mapping from T to $(\mathcal{I}^{\oplus})^P$,*
- *Post, the forward incidence mapping, is a mapping from T to $(\mathcal{I}^{\oplus})^P$,*
- *$\lambda : T \to \Sigma_\varepsilon$ is a labelling function.*

Since $(\mathcal{I}^{\oplus})^P$ is isomorphic to $(P \times \mathcal{I})^{\oplus}$, $\mathsf{Pre}(t)$ and $\mathsf{Post}(t)$ may also be considered as multisets. Given a place p and a transition t, if the multiset $\mathsf{Pre}(t)(p)$ (resp. $\mathsf{Post}(t)(p)$) is non null then it defines a *pre-arc* (resp. *post-arc*) of t connected to p.

A *configuration* ν of a *TdPN* is an item of $(\mathbb{R}^{\oplus}_{\geq 0})^P$ (or equivalently $(P \times \mathbb{R}_{\geq 0})^{\oplus}$). Intuitively, a configuration is a marking extended with age information for the tokens. We will write (p, τ) for a token which is in place p and whose age is τ. A configuration is then a finite sum of such pairs. A token (p, τ) then belongs to the configuration ν whenever $(p, \tau) \leq \nu$ (in terms of multisets). Given a configuration $\nu \in (P \times \mathbb{R}_{\geq 0})^{\oplus}$ and a multiset $f \in (P \times \mathcal{I})^{\oplus}$, we say that ν satisfies f, and write $\nu \models f$, if and only if there exists a multiset $x \in (P \times \mathbb{R}_{\geq 0} \times \mathcal{I})^{\oplus}$ verifying the following conditions.

$$\begin{cases} \pi_{1,2}(x) = \nu, \\ \pi_{1,3}(x) = f, \\ \forall (p, \tau, I) \in sup(x),\ \tau \in I. \end{cases}$$

(where $\pi_{i,j}$ denotes the projection on the ith and jth components of the tuples of the bag)

We now describe the semantics of a *TdPN* as a transition system.

Definition 4 (Semantics of a *TdPN*). *Let $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \lambda)$ be an TdPN. Its semantics is the transition system $(Q, \Sigma_\varepsilon, \to)$ where $Q = (\mathbb{R}^{\oplus}_{\geq 0})^P$ and the transition relation $\to$ is composed of delay and discrete transitions as follows:*

- *For each $d \in \mathbb{R}_{\geq 0}$, there is a delay transition $\nu \xrightarrow{d} \nu + d$ where the configuration $\nu + d$ is defined by $(\nu + d)(p) = \nu(p) + d$ for every $p \in P$.*
- *Given a transition $t \in T$ and two configurations $\nu, \nu' \in (P \times \mathbb{R}_{\geq 0})^{\oplus}$, there exists a discrete transition from ν to ν' labelled by $\lambda(t)$, denoted by $\nu \xrightarrow{\lambda(t)} \nu'$, if and only if there exist two multisets ${}^\bullet\nu, \nu^\bullet \in (P \times \mathbb{R}_{\geq 0})^{\oplus}$ such that:*

$$\begin{cases} {}^\bullet\nu \models \mathsf{Pre}(t), \\ \nu^\bullet \models \mathsf{Post}(t), \\ {}^\bullet\nu \leq \nu, \\ \nu' = \nu - {}^\bullet\nu + \nu^\bullet. \end{cases}$$

The intuition of the previous definition is as follows: ${}^\bullet\nu$ is the multiset of tokens which is removed from the configuration ν when firing transition t, whereas $\nu^\bullet$ is the multiset of tokens that are created by the transition firing. Moreover, the ages of all these tokens need to satisfy the constraints specified by the various arcs (conditions written using the $\models$ operator defined above). Finally, the new configuration is given by ν' computed as $\nu' = \nu - {}^\bullet\nu + \nu^\bullet$.

A *path* in the $TdPN\mathcal{N}$ is a sequence $\nu_0 \xrightarrow{d_1} \nu'_1 \xrightarrow{t_1} \nu_1 \xrightarrow{d_2} \nu'_2 \xrightarrow{t_2} \nu_2 \ldots$ in the above transition system, which alternates between delay and discrete transitions. A *timed transition sequence* is a finite timed word over alphabet T, the set of transitions of $\mathcal{N}$. A *firing sequence* is a timed transition sequence $(t_1, \tau_1)(t_2, \tau_2)\ldots$ such that $\nu_0 \xrightarrow{\tau_1} \nu'_1 \xrightarrow{t_1} \nu_1 \xrightarrow{\tau_2 - \tau_1} \nu'_2 \xrightarrow{t_2} \nu_2 \ldots$ is a path. If $(p, \tau) \leq \nu$ is a token of a configuration ν, it is a *dead token* whenever for every interval I labelling a pre-arc of p, τ is strictly greater than I. It means that this token cannot be used anymore by a pre–arc to fire a transition.

The untimed word which is read along a path $\nu_0 \xrightarrow{d_1} \nu'_1 \xrightarrow{t_1} \nu_1 \xrightarrow{d_2} \nu'_2 \xrightarrow{t_2} \nu_2 \ldots$ is the projection over Σ of the timed word i.e. $\lambda(t_1)\lambda(t_2)\ldots$ Timed Petri nets can be considered as language acceptors, as formally defined by the next definition. Observe that we require that initial and final configurations have *rational* values.

Definition 5 (Coverability Language accepted by a $TdPN$).
Let $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \lambda)$ be a $TdPN$ and ν_i, ν_f be two configurations with rational values. A finite path in $\mathcal{N}$ is accepting if it starts from ν_i and ends in a configuration $\nu \geq \nu_f$. We note $\mathcal{L}(\mathcal{N}, \nu_i, \nu_f)$ the set of finite untimed words accepted by $(\mathcal{N}, \nu_i, \nu_f)$ along finite paths.

The key observation is that w.r.t. the coverability language of a $TdPN$ (as in timed automata), it is sufficient to look for an abstraction of the configuration. In the next definition, max denotes the smallest integer greater or equal than the (finite) bounds of intervals of the net and the ages of tokens in ν_i, ν_f.

Definition 6 (Regions of $TdPN$s). *A region $\mathcal{R}$ for a tuple $(\mathcal{N}, \nu_i, \nu_f)$ is a sequence $a_0 a_1 \ldots a_n a_\infty$ where $n \in \mathbb{N}$, for all $0 < i \leq n$, $a_i \in (P \times \{0, 1, \ldots, \max - 1\})^\oplus$, $a_0 \in (P \times \{0, 1, \ldots, \max\})^\oplus$, and $a_\infty \in (P \times \{\infty\})^\oplus$ with $\mathsf{size}(a_i) \neq 0$ if $i \neq 0$.*

We informally explain the semantics of a region. Given the multiset of tokens defining a configuration, we obtain its associated region as follows. We put in a_∞ all the tokens whose ages are strictly greater than max and forget their ages. We then put in a_0 the tokens with integral ages and add the information about their ages. Finally, we order the remaining tokens depending on the fractional part of their ages in $a_1, \ldots, a_n$, forget their fractional part, and only store the integral part of their ages. Hence n is the number of different positive fractional values for ages of the remaining tokens. For instance, consider the multiset of tokens $(p, 1) + (p, 2.8) + (q, 0.8) + (q, 5.1) + (r, 1.5)$. Then, if the maximal constant is 4, its region encoding will be $a_0 a_1 a_2 a_\infty$ where $a_0 = (p, 1)$ (because there is a single token with integral age), $a_\infty = (q, \infty)$ (because the age of token $(q, 5.1)$ is 5.1, hence above the maximal constant), $a_1 = (r, 1)$ (among all fractional parts,

0.5 is the smallest one), and $a_2 = (p, 2) + (q, 0)$ (all tokens with fractional part 0.8).

Furthermore we can define an (infinite but countable) transition system over regions that generate the untimed words of the net. Rather than giving a formal cumbersome definition. We informally present it:

- We associate silent transitions with time elapsing. Since we can split the time elapsing, we consider two kinds of such transitions.
  1. Given a region $a_0 a_1 \dots a_n a_\infty$, when $a_0 > 0$ we first partition $a_0 = b_0 + c_0$ where b_0 (resp. c_0) is the multiset of tokens with age strictly less than max (resp. equal to max). The new region is now the word $0 b_0 a_1 \dots a_n (a_\infty + c_0)$. This transition corresponds to a small time elapsing that does not let the ages of tokens of a_n to reach or overcome an integral value.
  2. Given a region $0 a_1 \dots a_n a_\infty$ when $n > 0$, the new region is now the word $b_0 a_1 \dots a_{n-1} a_\infty$ where the tokens of b_0 are the tokens of a_n with their fractional part cancelled and their integral part increased by one. This transition corresponds to the time elapsing that lets the ages of tokens of a_n reach an integral value.
- The information associated with the age of tokens in a region is sufficient to know whether they belong to an interval labelling a pre-arc. So given a region $a_0 a_1 \dots a_n a_\infty$, in order to fire t:
  1. We must constitute a word $b_0 b_1 \dots b_n b_\infty$ with $b_i \leq a_i$ for every $i \in \{0, \dots, n\} \cup \infty$ such that for every place p there is a bijective mapping from the intervals of the multiset $\mathsf{Pre}(p, t)$ to the tokens labelled by p in $b_0, b_1, \dots b_n, b_\infty$. The first step of the firing consists then to delete these tokens leading to an intermediate region $c_0 c_1 \dots c_\infty = (a_0 - b_0)(a_1 - b_1) \dots (a_n - b_n)(a_\infty - b_\infty)$ where the the c_i's for $1 \leq i \leq n'$such that $c_i = 0$ are then deleted.
  2. Then for every place p and every interval of multiset $\mathsf{Post}(p, t)$, we choose a token whose fractional part may be either null, either a non null existing one or a new non null one, in this last case increasing n and choosing any position in the fractional order. The choice must lead to an age belonging the interval. These new tokens "added" to $c_0 c_1 \dots c_n c_\infty$ lead to the region reached by this firing of t (as there are non deterministic choices, several but finite firings of t are possible).

4 Equivalence between PDNs and TdPNs

Theorem 1. *Let $\mathcal{N}$ be a PDN and m_i, m_f be two markings of $\mathcal{N}$. Then there exist $\mathcal{N}'$ a TdPN and two rational configurations ν_i, ν_f such that $\mathcal{L}(\mathcal{N}', \nu_i, \nu_f) = L(\mathcal{N}, m_i, m_f)$.*

Proof. Let us first describe the principles of the simulation. Places of $\mathcal{N}'$ will contain two kinds of tokens: the tokens of age belonging to $[0, 1]$ will be *relevant* while the older tokens will be irrelevant. We define the relevant part of a marking of $\mathcal{N}'$ as the marking where the irrelevant tokens have been deleted.

The simulation of a transition firing will last 1 time unit (t.u.). So the markings of $\mathcal{N}'$ at instants $0, 1, \ldots$ are the basis of the simulation.

Our simulation is lossy in the following sense. Assume there is a firing sequence $m_i \xrightarrow{\sigma} m$ in $\mathcal{N}$, then there is at least one *perfect simulation* $\nu_i \xrightarrow{\sigma} \nu$ in $\mathcal{N}'$ with the same associated word. Furthermore all firing sequences of $\mathcal{N}'$ will be perfect or lossy simulations. A *lossy simulation* is a sequence that leads to markings at integer instants whose relevant parts are covered by the relevant part of markings reached a perfect simulation with the same associated word.

For technical reasons, a place p of $\mathcal{N}$ will be simulated by two places p_0, p_1 of $\mathcal{N}'$. Let $m_i \xrightarrow{\sigma} m$ be a firing sequence in $\mathcal{N}$, with n current identities $x^1 < \ldots < x^n$ in m and denote m by the word $(\sum_{p\in P} \lambda_p^1 \cdot p) \cdots (\sum_{p\in P} \lambda_p^n \cdot p)$.

Let $\nu_i \xrightarrow{\sigma} \nu$ be some perfect simulation of σ in $\mathcal{N}'$. There will be exactly n fractional parts of ages of relevant tokens in ν. Assume that the length of the firing sequence σ is even (resp. odd). Let us denote $a_0 a_1 \ldots a_n a_\infty$ be the region associated with ν. Then the word a_i fulfills $a_i = \sum_{p\in P} \lambda_p^i \cdot (p_0, 0)$ (resp. $a_i = \sum_{p\in P} \lambda_p^i \cdot (p_1, 0)$). a_0 will contain tokens of control places (to be detailed later) and a_∞ will equal to 0.

Let us describe the control places,

- $time_0, time_1$ are the places that schedule the operations. At an even (resp. odd) instant place $time_0$ (resp. $time_1$) has a token with age 0. Then after one t.u., a transition tt_0 (resp. tt_1) ending the simulation process is fired getting this token and producing a token with age 0 in $time_1$ (resp. $time_0$).
- Place $idle_0$ (resp. $idle_1$) has a token only present at even (resp. odd) instants. The consumption of this token by transition t_0 (resp. t_1) starts the simulation process of transition t. When a simulation is started, a token (whose time is irrelevant) is produced in place $trans_0$ (resp. $trans_1$). This token enables to transfer relevant tokens that *will not be used* in the transition firing. When such a token (say in place q_0) has age 1 it is consumed by $tr.q_0$ and a token is produced in q_1. At the end of the simulation the token has the same age as the original one at the beginning of the simulation. Observe that some tokens may be *forgotten* (case of a lossy simulation). These forgotten tokens cannot be used in the sequel since their age become greater than 1.
- Let us recall that all variables occurring in a transition t of $\mathcal{N}$ are totally ordered. Thus the transition simulation consumes and produces the tokens required by variables, beginning by the greatest variable. Let us illustrate this simulation in the example of figure 2. The token "with identity Z" in place q which must be consumed will be the first one to reach age 1, so it is deleted by transition $simZ.t_0$. Then transition $simY.t_0$ produces the token "with identity Y" in place s_1. Finally, transition $simX.t_0$ consumes the token "with identity X" in place p_0 and simultanously produces the token in place r_1. Observe that these transitions must let time elapse due to the interval constraints. This avoids to use the same identity for X, Y and Z.

Let us now explain by an example (see figure 3) how to check coverability of marking $p(q + r)$ in $\mathcal{N}$. At even (resp. odd) instants one consumes the token in

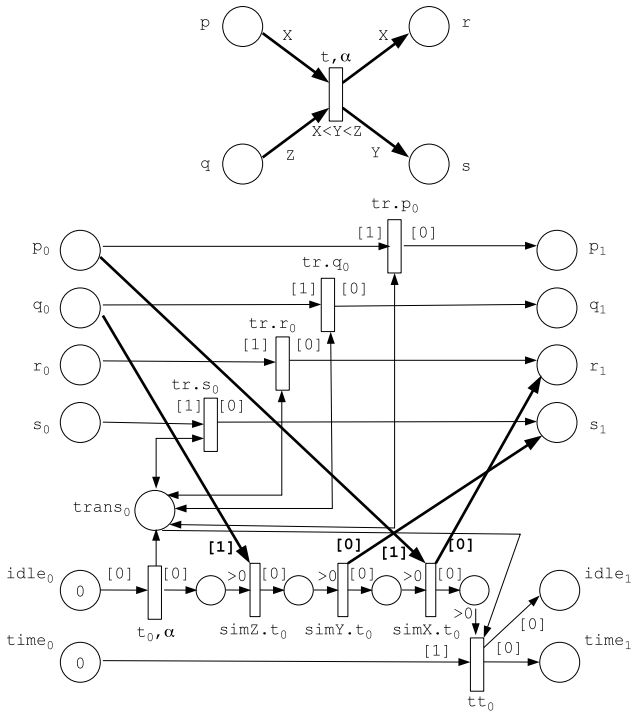

Fig. 2. Simulation of a transition at even instants

$idle_0$ (resp. $idle_1$) and proceeds to test the coverability. First one lets time elapse until we obtain a token in q and r with age 1. Then after some time elapsing we must obtain a token in p with age 1 and we conclude positively by covering place *success*. The generalization is straightforward.

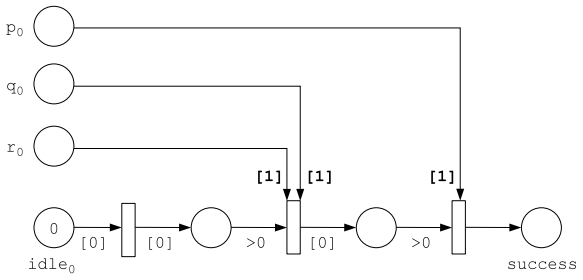

Fig. 3. Test of covering $p(q + r)$ at even instants

Thus the final marking is defined by $\nu_f = 1 \cdot (success, 0)$. Let us define ν_i. Assume there are n identities in m_i, then we choose n rational values in $(0, 1)$ and mark the places p_0 with $p \in P$ according to $m_i(p)$. Finally we add a token with age 0 in $idle_0$ and $time_0$.

In order to prove the reverse implication, we recall that for any $TdPN\mathcal{N}$ there is a $TdPN\mathcal{N}'$ with same language and such that the only interval occurring in post-arcs is $[0, 0]$ (theorem 4 of [5]). Furthermore since ν_i and ν_f have rational values and we consider untimed languages, then by changing the time scale we assume w.l.o.g. that ν_i and ν_f have integer values (less or equal than the max value associated with regions).

Theorem 2. *Let $\mathcal{N}$ be a TdPN of $\mathcal{N}$ and ν_i, ν_f be two integer configurations. Then there exists $\mathcal{N}'$ a PDN and m_i, m_f two markings such that $L(\mathcal{N}', m_i, m_f) = \mathcal{L}(\mathcal{N}, \nu_i, \nu_f)$.*

Proof. As in the previous proof, our simulation is a lossy simulation allowing to "loose" tokens of net $\mathcal{N}$ in the simulating net $\mathcal{N}'$ as it does not change the covering language. We first describe the principle of the simulation.

After some initialization stage, places $low, high$ and int always contain a single token. In the sequel of proof we denote the identity contained in such a place by the name of the place. Every non null fractional part of the current configuration of $\mathcal{N}$ is represented by an identity x such that: $low \leq x \leq high$. The order of such identities is the reverse order of the fractional part: for two identities of fractional parts $x < y$ the fractional part of x is greater than the fractional part of y. For every simulation, low is just a lower bound of the identity with the

highest fractional part but if there is at least a token in $\mathcal{N}$ whose age and has a non null fractional part then there is always a simulation for which *low* is equal to this identity. Furthermore, identity *int* corresponds to tokens whose age (less or equal than max) have a null fractional part.

During the simulation, *int* only decreases while *low* only increases, and as in the initial marking we ensure that $int < low$, this inequation will always be fulfilled. At any instant of the simulation, the identities that label tokens have identities between *int* and *high* and only tokens which have identities between *low* and *high* or equal to *int* are still relevant for the simulation.

Let max be the maximal constant as defined above for $\mathcal{N}$, ν_i and ν_f. For every place p of $\mathcal{N}$, $\mathcal{N}'$ has the following places: $p_0, p_1, \ldots, p_{\max}, p_\infty$. Place p_k contains the tokens of $\mathcal{N}$ in p with age less or equal than max and integral part equal to k. Place p_∞ contains the tokens of $\mathcal{N}$ in p with age greater than max; this place contains black tokens as the fractional part of the age is irrelevant for such tokens in $\mathcal{N}$ are irrelevant.

During the simulation, place *disc* is either empty or contains a black token that allows the simulation of discrete transitions of $\mathcal{N}$. Let us first describe the simulation of time elapsing as illustrated in figure 4. Transition el_0 begins to perform the simulation of a small elapse of time whose only effect (see above) is that there is no more tokens (with age less or equal than max) with integral ages. It increases *high* in order to assign this value to the tokens with the integral ages. While $time_0$ is marked, transition $tf_{p,k}$ with $k <$ max "updates" tokens with integer age in p_k changing their identity to *high*. Transition $tf_{p,\max}$ transfers tokens with age max from p_k to p_∞. As said before, some tokens can be forgotten but they will not perturb the simulation since at the end of the transfer *int* is decreased (transition el_1). Then either we stop time elapsing simulation (transition el_2) or proceed (transition el_3) to let an additional amount of time that corresponds to let the tokens with greatest fractional part reach their next integral value by changing their identity from *low* to *int* and moving tokens from p_k to p_{k+1}. When *low* is different from the identity of tokens with greatest fractional part, no transfer occur. At the end of a simulation, a new value is chosen for *low* greater than the former value and less or equal than *high* (transition el_4). When this choice corresponds to the identity of the new greatest fractional part the simulation is exact. Otherwise, the tokens whose fractional parts have associated identities less than *low* are "lost".

The simulation of a transition of $\mathcal{N}$ is straightforward. In order to simplify its presentation, we can assume w.l.o.g. that pre-arcs are labelled by multisets of intervals $[0],]0,1[, [1], \ldots, [\max],]\max, \infty[$. This can be easily obtained by duplicating transitions (see for instance [5]). As said before, post-arcs are labelled by a multiset over interval [0]. Rather than defining it formally we illustrate the translation on figure 5. For instance, since the arc from p to t is labelled by $]2,3[$, we are looking for a token in place p_2 with identity between *low* and *high*. The other cases are similar. Observe that since post-arcs are labelled by 0, there is no new fractional part. This avoids to handle the undesirable case where a new

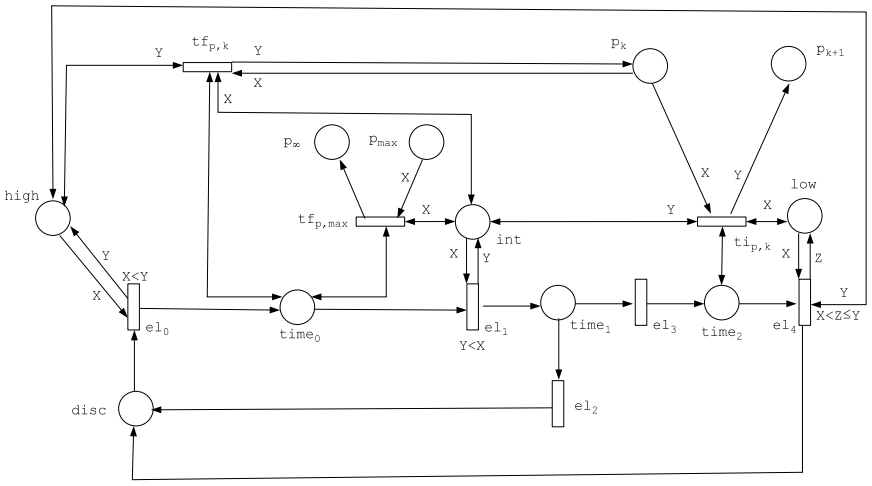

Fig. 4. Simulation of time elapsing

fractional part would be the greatest one, as it would require to decrease *low* which is forbidden by our simulation.

Checking the coverability condition is performed by a transition *stop* that consumes the token in *disc* and consumes tokens $n_{p,k}$ tokens in place p_k with identity *int* where $n_{p,k}$ is the number of tokens of $\nu_f(p)$ with age k. This transition produces a token with any identity in a place *success*. Thus $m_f = success$.

The initial marking m_i is defined by choosing two identities say $id < id'$ marking place *int* with a *id*-token and places *low* and *high* with a id'-token. Place *disc* is marked with a black token. Finally one marks places p_k with $m_{p,k}$ *id*-tokens where $m_{p,k}$ is the number of tokens in $\nu_i(p)$ with age k.

Combining theorems 1 and 2, we obtain the desired result.

Theorem 3. *PDN* $\simeq$ *TdPN*

References

1. P.A. Abdulla, G. Delzanno, and L. Van Begin. Comparing the Expressive Power of Well-Structured Transition Systems. 21st Int. Workshop on Computer Science Logic, CSL'07, LNCS vol. 4646, pp. 99-114. Springer, 2007.
2. P.A. Abdulla, G. Delzanno, and L. Van Begin. A Language-Based Comparison of Extensions of Petri Nets with and without Whole-Place Operations. LATA'09, LNCS vol. 5457, pp. 71-82. Springer, 2009.

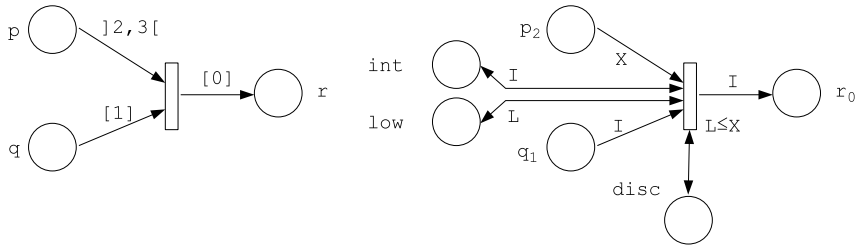

Fig. 5. Simulation of a transition

3. P.A. Abdulla and A. Nylen. Timed Petri Nets and BQOs. 22nd Int. Conf. on application and theory of Petri nets, ICATPN'01 LNCS vol. 2075, pp. 53-70 Springer, 2001.
4. R. Alur, C. Courcoubetis, and T. A. Henzinger. The Observational Power of Clocks. Proceedings of the Fifth International Conference on Concurrency Theory (CONCUR), LNCS 836, Springer, 1994, pp. 162-177.
5. P. Bouyer, S. Haddad and P.-A. Reynier. Timed Petri Nets and Timed Automata: On the Discriminating Power of Zeno Sequences. Information and Computation 206(1), pages 73-107, 2008.
6. A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. 14th Int. Colloquium on Automata, Languages and Programming, ICALP'87, LNCS vol. 267, pp. 499-508. Springer, 1987.
7. A. Finkel, P. McKenzie, and C. Picaronny. A well-structured framework for analysing Petri net extensions. Information and Computation, vol. 195(1-2):1-29 (2004).
8. G. Geeraerts, J. Raskin, and L. Van Begin. Well-structured languages. Acta Informatica, 44:249-288. Springer, 2007.
9. R. Lazic, T.C. Newcomb, J. Ouaknine, A.W. Roscoe, and J. Worrell. Nets with Tokens Which Carry Data. Fund. Informaticae 88(3):251-274. IOS Press, 2008.
10. F. Rosa-Velardo and D. de Frutos-Escrig. Name creation vs. replication in Petri Net systems. *Fund. Informaticae* 88(3). IOS Press (2008) 329-356.
11. F. Rosa-Velardo, and G. Delzanno. Language-Based Comparison of Petri Nets with black tokens, pure names and ordered data. LATA'10, LNCS vol. 6031, pp. 524-535. Springer, 2010.