Comparing Petri Data Nets and Timed Petri Nets

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

December 2010

Research report LSV-10-23
Comparing Petri Data Nets  
and Timed Petri Nets

R. Bonnet¹, A. Finkel¹, S. Haddad¹*, F. Rosa-Velardo²**

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 expressivity.  
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 DE-SAFIOS10 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 \(\nu\)-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 \((x, y) \leq_{X \times Y} (x', y') \iff x \leq_X x' \land y \leq_Y y'\). Their disjoint union, denoted \(X \cup Y\) is well ordered by:

\[
   z \leq_{X \cup Y} z' \iff \begin{cases} z, z' \in X \land z \leq_X z' \lor z, z' \in Y \land z \leq_Y z' 
\end{cases}
\]

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 \(<_{iex} \) in \(\bigcup_k X_1 \times \ldots \times X_k\) by \((x_1, \ldots, x_p) <_{iex} (x'_1, \ldots, x'_q)\) iff there is \(i \in \{1, \ldots, \min(p, q)\}\) such that \(x_j = x'_j\) for \(j < i\) and \(x_i <_{iex} x'_i\) or \((x_1, \ldots, x_p) = (x'_1, \ldots, x'_q)\).
Given a partial function (shortly: function) \( f : X \rightarrow Y \), the domain of \( f \) is defined by \( \text{dom}(f) = \{ x \in X \mid \exists y \in Y \text{ such that } f(x) = y \} \) and its range by \( \text{range}(f) = \{ y \in B \mid \exists x \in A \text{ such that } f(x) = y \} \). A function \( f \) is total if \( \text{dom}(f) = X \).

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 \( \text{range}(f) = Y \). Finally, if \( X \) and \( Y \) are quasi-ordered (shortly: ordered), \( f \) is monotonic if \( x \leq y \implies f(x) \leq f(y) \).

It is an order embedding (shortly: embedding) if \( \varphi(x) \leq \varphi(x') \iff x \leq x' \). 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 \( m : X \rightarrow \mathbb{N} \) with a finite support \( \text{sup}(m) = \{ x \in X \mid m(x) \neq 0 \} \). We use the set-like notation \( \{m(x)\} \) 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 \( \{x_1, \ldots, x_n\} \leq_{\oplus} \{x'_1, \ldots, x'_{n'}\} \) if there is an injection \( h : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\} \) 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 \( \epsilon \). 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: \( x_1 \cdots x_n \leq_X x'_1 \cdots x'_{m'} \) if there is a strictly increasing mapping \( h : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\} \) such that \( x_i \leq x'_{h(i)} \) for each \( i \in \{1, \ldots, n\} \).

**WSTS.** A labelled transition system is a tuple \( \mathcal{S} = \langle X, \Sigma, \rightarrow \rangle \) where \( X \) is the set of states, \( \Sigma \) is the labelling alphabet and \( \rightarrow \subseteq X \times (\Sigma \cup \{\epsilon\}) \times X \) is the transition relation.

We write \( x \xrightarrow{a} x' \) to say that \( (x, a, x') \in \rightarrow \). This relation is extended for \( u \in \Sigma^* \) by \( x \xrightarrow{u} x' \iff x \xrightarrow{a_1} x_1 \cdots x_{k-1} \xrightarrow{a_k} x' \) and \( u = a_1 a_2 \cdots a_k \) (note that some \( a_i \)'s can be equal to \( \epsilon \)). A well structured transition system (shortly a WSTS) is a tuple \( \mathcal{S} = \langle X, \Sigma, \rightarrow, \leq_X \rangle \), where \( (X, \Sigma, \rightarrow) \) is a labelled transition system, and \( \leq_X \) is a wqo on \( X \), satisfying the following monotonicity condition: for all \( x_1, x_2, x'_1 \in X, x_1 \leq x'_1, x_1 \xrightarrow{u} x_2 \) 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 \( X \) of wqos, we will denote by \( \text{WSTS}_X \) the class of WSTS with state space in \( X \), or just \( \text{WSTS}_X \) for \( \text{WSTS}_X \).

**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 \( \omega \)-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, \( S_1 \) and \( S_2 \), we write \( S_1 \preceq S_2 \) whenever for every language \( L \) recognized by a system \( \Xi^1 \in S_1 \), there exists another system \( \Xi^2 \in S_2 \) and two states \( s_1, s_f \) such that \( L(\Xi^2, s_1, s_f) = L \). When \( S_1 \preceq S_2 \) and \( S_2 \preceq S_1 \), one denotes the equivalence of classes by \( S_1 \simeq S_2 \).

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 0 the null vector in any \( \mathbb{N}^k \) and for a word \( w = x_1 \ldots x_n \), we write \( |w| = n \) and \( w(i) = x_i \). \( \Sigma \) is a finite alphabet and \( \Sigma_\varepsilon \) denotes the alphabet enlarged with the empty word \( \varepsilon \).

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 \( t \in T \), \( F_t, H_t \in (\mathbb{N}^{|P|})^+ \) 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 \( \mathbb{N}^{|P|} \setminus 0 \). A sequence \( s' \in (\mathbb{N}^{|P|})^+ \) is a 0-extension of a marking \( s \) (or \( s \) is the 0-contraction of \( s' \)) if \( s \) can be obtained by removing all the occurrences of 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 0-contraction of \( u_0 x_1 u_1 \cdots u_{n-1} x_n u_n \) with \( u_i \in (\mathbb{N}^{|P|} \setminus \mathbf{0})^* \) and \( x_i \in \mathbb{N}^{|P|} \).

\(- x_i \geq F_i(i) \) and \( y_i = (x_i - F_i(i)) + H_i(i) \) for \( i \in \{1, \ldots, n\} \).

\(- s' \) is the 0-contraction of \( u_0 y_1 u_1 \cdots u_{n-1} y_n u_n \) 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 \( |F_i| = |H_i| = 2 \), and if \( F_i(1) = (n_1, n_2) \) 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, \( F_i(2) = (m_1, m_2) \) specifies the number of tokens that carry the datum \( d_y > d_x \) to which \( Y \) is instantiated that have to be removed (and similarly for \( H_i \)). The corresponding labels would be \( n_1 X + m_1 Y \) (resp. \( n_2 X + m_2 Y \)) 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

\[ p \xrightarrow{[2,3]+[0]} t \xrightarrow{0,3+[2,\infty]} q \]

**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 TdPNs. In the sequel, \( \mathcal{I} \) is the set of intervals with bounds in \( \mathbb{N} \cup \{\infty\} \).

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

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

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

A configuration \( \nu \) 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 \( (p, \tau) \) for a token which is in place \( p \) and whose age is \( \tau \). A configuration is then a finite sum of such pairs. A token \( (p, \tau) \) then belongs to the configuration \( \nu \) 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 \( \nu \) 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:

\[
\begin{align*}
\pi_{1,2}(x) &= \nu, \\
\pi_{1,3}(x) &= f, \\
\forall (p, \tau, I) \in \text{sup}(x), \tau \in \mathcal{I}.
\end{align*}
\]

(where \( \pi_{i,j} \) denotes the projection on the \( i \)th and \( j \)th 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 \( \mathcal{N} = (P, T, \text{Pre}, \text{Post}, \lambda) \) be an TdPN. Its semantics is the transition system \( (Q, \Sigma_e, \rightarrow) \) where \( Q = (\mathbb{R}^{\oplus}_{\geq 0})^P \) and the transition relation \( \rightarrow \) 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 \( \nu \) to \( \nu' \) labelled by \( \lambda(t) \), denoted by \( \nu \xrightarrow{\lambda(t)} \nu' \), if and only if there exist two multisets \( \bullet \nu, \nu' \in (P \times \mathbb{R}_{\geq 0})^{\oplus} \) such that:

\[
\begin{align*}
\bullet \nu &\models \text{Pre}(t), \\
\nu' &\models \text{Post}(t), \\
\bullet \nu &\leq \nu, \\
\nu' &= \nu - \bullet \nu + \nu'.
\end{align*}
\]
The intuition of the previous definition is as follows: $\bullet \nu$ is the multiset of tokens which is removed from the configuration $\nu$ 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 $\nu'$ 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 $\nu_0 \xrightarrow{t_1} \nu_1' \xrightarrow{\tau_1} \nu_1 \xrightarrow{t_2} \nu_2' \xrightarrow{\tau_2} \nu_2 \ldots$ is a path. If $(p, \tau) \leq \nu$ is a token of a configuration $\nu$, it is a dead token whenever for every interval $I$ labelling a pre-arc of $p$, $\tau$ 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 $\Sigma$ 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 $\mathcal{N} = (P, T, \text{Pre}, \text{Post}, \lambda)$ be a TdPN and $\nu_i, \nu_f$ be two configurations with rational values. A finite path in $\mathcal{N}$ is accepting if it starts from $\nu_i$ 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 $\nu_i, \nu_f$.

**Definition 6 (Regions of TdPNs).** A region $R$ for a tuple $(\mathcal{N}, \nu_i, \nu_f)$ is a sequence $a_0a_1\ldots a_n a_\infty$ where $n \in \mathbb{N}$, for all $0 < i \leq n$, $a_i \in (P \times \{0, 1, \ldots, \text{max} - 1\})^\oplus$, $a_0 \in (P \times \{0, 1, \ldots, \text{max}\})^\oplus$, and $a_\infty \in (P \times \{\infty\})^\oplus$ with $\text{size}(a_i) \neq 0$ 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 $a_\infty$ 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 $a_0a_1a_2a_\infty$ 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 \( a_0a_1 \ldots a_n a_\infty \), 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 \( \theta b_0a_1 \ldots a_n (a_\infty + c_0) \). 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 \( 0a_1 \ldots a_n a_\infty \) when \( n > 0 \), the new region is now the word \( b_0a_1 \ldots a_{n-1} a_\infty \) 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 \( a_0a_1 \ldots a_n a_\infty \), in order to fire \( t \):
  1. We must constitute a word \( b_0b_1 \ldots b_n b_\infty \) with \( b_i \leq a_i \) for every \( i \in \{0, \ldots, n\} \cup \infty \) such that for every place \( p \) there is a bijective mapping from the intervals of the multiset \( \text{Pre}(p, t) \) to the tokens labelled by \( p \) in \( b_0, b_1, \ldots, b_n, b_\infty \). The first step of the firing consists then to delete these tokens leading to an intermediate region \( c_0c_1 \ldots c_\infty = (a_0 - b_0)(a_1 - b_1) \ldots (a_n - b_n)(a_\infty - b_\infty) \) 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 \( \text{Post}(p, t) \), 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 \( c_0c_1 \ldots c_n c_\infty \) 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 \( \nu_i, \nu_f \) such that \( L(\mathcal{N}', \nu_i, \nu_f) = L(\mathcal{N}, m_i, m_f) \).

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 \( N' \) at instants 0, 1, ... are the basis of the simulation.

Our simulation is lossy in the following sense. Assume there is a firing sequence \( m_i \rightarrow m \) in \( N \), then there is at least one perfect simulation \( \nu_i \rightarrow \nu \) in \( N' \) with the same associated word. Furthermore all firing sequences of \( N' \) 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 \( N \) will be simulated by two places \( p_0, p_1 \) of \( N' \). Let \( m_i \rightarrow m \) be a firing sequence in \( N \), with \( n \) current identities \( x^1 < \ldots < x^n \) in \( m \) and denote \( m \) by the word \( (\sum_{p \in P} \lambda^1_p \cdot p) \cdots (\sum_{p \in P} \lambda^n_p \cdot p) \).

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

Let us describe the control places,

- \( \text{time}_0, \text{time}_1 \) are the places that schedule the operations. At an even (resp. odd) instant place \( \text{time}_0 \) (resp. \( \text{time}_1 \)) has a token with age 0. Then after one t.u., a transition \( t_0 \) (resp. \( t_1 \)) ending the simulation process is fired getting this token and producing a token with age 0 in \( \text{time}_1 \) (resp. \( \text{time}_0 \)).
- \( \text{Place idle}_0 \) (resp. \( \text{idle}_1 \)) 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 \( \text{trans}_0 \) (resp. \( \text{trans}_1 \)). 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 \( \text{tr} \cdot q_0 \) 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 \( N \) 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 \( \text{sim}Z \cdot t_0 \). Then transition \( \text{sim}Y \cdot t_0 \) produces the token “with identity \( Y \)” in place \( s_1 \). Finally, transition \( \text{sim}X \cdot t_0 \) consumes the token “with identity \( X \)” in place \( p_0 \) and simultaneously 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 \( N \). 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.

Thus the final marking is defined by $\nu_f = 1 \cdot (success, 0)$. Let us define $\nu_i$. 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 $T_{dPN}N'$ there is a $T_{dPN}N'$ with same language and such that the only interval occurring in post-arcs is $[0, 0]$ (theorem 4 of [5]). Furthermore since $\nu_i$ and $\nu_f$ have rational values and we consider untimed languages, then by changing the time scale we assume w.l.o.g. that $\nu_i$ and $\nu_f$ have integer values (less or equal than the max value associated with regions).

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

**Proof.** As in the previous proof, our simulation is a lossy simulation allowing to “loose” tokens of net $N$ in the simulating net $N'$ 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 $N$ 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 \( \mathcal{N} \), \( \nu_1 \) and \( \nu_f \). For every place \( p \) of \( \mathcal{N} \), \( \mathcal{N}' \) has the following places: \( p_0, p_1, \ldots, p_{\text{max}}, p_{\infty} \). Place \( p_k \) contains the tokens of \( \mathcal{N} \) in \( p \) with age less or equal than max and integer part equal to \( k \). Place \( p_{\infty} \) 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 elo 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 timeo is marked, transition \( tf_{p,k} \) with \( k < \text{max} \) “updates” tokens with integer age in \( p_k \) changing their identity to high. Transition \( tf_{p,\text{max}} \) transfers tokens with age max from \( p_k \) to \( p_{\infty} \). 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 el1). Then either we stop time elapsing simulation (transition el2) or proceed (transition el3) 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 el4). 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 \([0], [0, 1], [1], \ldots, [\text{max}], [\text{max}, \infty]\). 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 \( m_f = \text{success} \).

The initial marking \( m_i \) is defined by choosing two identities say \( id < id' \) marking place int with an \( id\)-token and places low and high with an \( id'\)-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**