International Journal of Foundations of Computer Science

CONTINUOUS PETRI NETS: EXPRESSIVE POWER AND DECIDABILITY ISSUES

LAURA RECALDE
GISED, University of Zaragoza, Mara de Luna, 1
50018 Zaragoza, Spain

and

SERGE HADDAD
LSV CNRS, ENS Cachan, 61, avenue du Président Wilson
94235 Cachan Cedex, France

and

MANUEL SILVA
GISED, University of Zaragoza, Mara de Luna, 1
50018 Zaragoza, Spain

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

State explosion is a fundamental problem in the analysis and synthesis of discrete event systems. Continuous Petri nets can be seen as a relaxation of the corresponding discrete model. The expected gains are twofold: improvements in complexity and in decidability. In the case of autonomous nets we prove that liveness or deadlock-freeness remain decidable and can be checked more efficiently than in Petri nets. Then we introduce time in the model which now behaves as a dynamical system driven by differential equations and we study it w.r.t. expressiveness and decidability issues. On the one hand, we prove that this model is equivalent to timed differential Petri nets which are a slight extension of systems driven by linear differential equations (LDE). On the other hand, (contrary to the systems driven by LDEs) we show that continuous timed Petri nets are able to simulate Turing machines and thus that basic properties become undecidable.

Keywords: Petri nets, Dynamical systems, Reachability, Liveness, Expressiveness, Decidability

1. Introduction

Autonomous continuous systems. State explosion problems represent a main drawback in the study of heavily loaded discrete event dynamic systems, modeled for example as Petri nets. Trying to alleviate these problems, different relaxations have been used. One main relaxation consists in considering "fractions"

of a (unit) event to occur. This naturally leads to a continuous state space (including the discrete state space). Furthermore such systems often allow a simpler introduction of continuous time in their behaviour.

Hybrid and (timed) continuous systems. Dynamic systems can be classified depending on the way time is represented. Generally, trajectories of discrete-time systems are obtained by iterating a transition function whereas the ones of continuous-time systems are often solutions of a differential equation. When a system includes both continuous and discrete transitions it is called a *hybrid system* [8]. A special kind of hybrid systems where the trajectories are continuous (w.r.t. standard topology) and right-differentiable functions of time have been intensively studied. They are defined by a finite number of regions and associated ordinary differential equations (ODEs) such that inside a region r, a trajectory fulfills the equation $\dot{x}_d = f_r(x)$ where x is the trajectory and $\dot{x}_d$ its right derivative. These additional requirements are not enough to limit their expressiveness. For instance, the model of [3] has piecewise constant derivatives inside regions which are polyhedra and it is Turing equivalent if its space dimension is at least 3.

Differentiable systems. A more stringent requirement consists in describing the dynamics of the system by a single ODE $\dot{x} = f(x)$ where f is continuous, thus yielding continuously differentiable trajectories. We call such models, *differentiable systems*. In [4], the author shows that differentiable systems in $\mathbb{R}^3$ can simulate Turing machines. The corresponding ODE is obtained by extrapolation of the transition function of the Turing machine over every possible configuration. Indeed such a configuration is represented as a point in the first dimension of the ODE (and also in the second one for technical reasons) and the third dimension corresponds to the time evolution. The explicit local ODE around every representation of a configuration is computed from this configuration and its successor by the Turing machine. Thus the explicit equations of the ODE are piecewise defined inside *an infinite number of regions* which is far beyond the expressiveness of standard ODE formalisms used for the design and analysis of dynamical systems. So the question to determine which (minimal) set of operators in an explicit expression of f is required to obtain Turing machine equivalence, is still open.

Our contribution. Our contribution is related to (timed) continuous Petri nets. Continuous Petri nets is the model obtained from Petri nets by allowing fractions of firings to occur. Several decidability results have been established for this model (see e.g. [10]). Here, we extend these results to two fundamental behavioural problems: the liveness and the deadlock freeness of a net.

Then we study two ways of introducing time in the model yielding to timed continuous Petri nets (TCPNs) and timed differential Petri nets (TDPNs). In the former model the firing speed of a transition is determined by the current marking of its input places together with the weight of the input arcs. In the latter model, an additional speed control matrix together with the global current marking determines this speed. We show that the two models are equivalent and exhibit optimal translations. We also point out simplified but equivalent models with only a linear increasing of the size of the model.

At last, we make more precise the boundary between decidability and undecidability in differentiable systems by showing that TDPNs can simulate Turing machines. Indeed the ODE ruling this model is particularly simple. First its expression is a linear expression enlarged with the "minimum" operator. Second, it can be decomposed into a finite number of linear LDEs $\dot{x} = \mathbf{M} \cdot x$ (with $\mathbf{M}$ a matrix) inside polyhedra. The other side of the boundary is given by the decidability of the reachability problem in systems of linear differentiable equations [7].

The structure of the paper[a] is as follows: in Section 2 (autonomous) continuous Petri nets are introduced with previous results and then the decidability of liveness or deadlock-freeness of the model is established. Section 3 deals with the timed model, proving that timed continuous Petri nets and timed differentiable Petri nets are equivalent with essentially optimal translations. Finally in Section 4, we establish that timed continuous Petri nets can simulate Turing machine and deduce undecidability of basic properties.

2. Continuous Petri Nets

2.1. Syntax and Semantics

We assume that the reader is familiar with Petri nets (PNs) (for notation we use the standard one, see for instance [13]). The structure of *(autonomous) continuous Petri nets* (CPNs not to be confused with coloured Petri nets) is the same as the structure of discrete PNs.

Definition 1 (CPN Syntax) $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ *a continuous Petri net is defined by :*

- P *a finite set of places,*
- T *a finite set of transitions with* $P \cap T = \emptyset$*,*
- **Pre** *and* **Post** *are* $|P| \times |T|$ *sized, natural valued,* pre- and post- incidence matrices.

Notations. $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ the token-flow matrix. ${}^\bullet t = \{p \in P \mid \mathbf{Pre}[p,t] > 0\}$ (resp. $t^\bullet = \{p \in P \mid \mathbf{Pre}[p,t] > 0\}$) is the set of input (resp. output) places of t. The notations are extended to sets by ${}^\bullet\Theta = \bigcup_{x \in \Theta} {}^\bullet x$ and $\Theta^\bullet = \bigcup_{x \in \Theta} x^\bullet$. Let $\mathbf{M}$ be $|P| \times |T|$ matrix then $\mathbf{M}[P,t]$ denotes the column vector of $\mathbf{M}$ indexed by t.

A marking $\mathbf{m}$ of a CPN is a non negative real amount of tokens $\mathbf{m}(p)$ in every place p, *i.e.* $\mathbf{m} \in (\mathbb{R}_{\geq 0})^{|P|}$. A marked CPN $\langle \mathcal{N}, \mathbf{m_0} \rangle$ is thus given by $\mathcal{N}$ a CPN equipped with $\mathbf{m_0}$ an initial marking. The usual PN system, $\langle \mathcal{N}, \mathbf{m_0} \rangle$, will be said to be *discrete* so as to distinguish it from a continuous PN system, in which $\mathbf{m_0} \in (\mathbb{R}_{\geq 0})^{|P|}$. The main difference between both formalisms is in the evolution rule, since in continuous PNs firing is not restricted to be done in integer amounts. As a consequence the marking is not forced to be integer. We could also consider the incidence matrices as rational valued without changing any of the results but we want to emphasize the fact that CPNs can be viewed as an abstraction of Petri nets obtained by relaxation.

[a] This paper combines the results presented in [6] and [11].

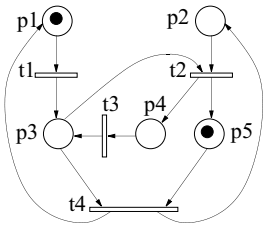

Figure 1: This system is not reversible as discrete, or as continuous with finite number of firings, but it is lim-reversible.

Definition 2 (CPN Semantics) *Let $\mathcal{N}$ be a continuous Petri net,* $\mathbf{m}$ *be marking of $\mathcal{N}$ and t be a transition. Then :*

- *The* enabling degree[b] *of t at* $\mathbf{m}$ *is* $\text{enab}(t, \mathbf{m}) = \min_{p \in {}^\bullet t}\{\mathbf{m}[p]/\mathbf{Pre}[p,t]\}$
- *t is* enabled *at* $\mathbf{m}$ *iff* $\text{enab}(t, \mathbf{m}) > 0$
- *If t is* enabled *at* $\mathbf{m}$ *then for every real α s.t. $0 < \alpha \leq \text{enab}(t, \mathbf{m})$ the α-firing of t leads to a new marking $\mathbf{m}' = \mathbf{m} + \alpha \cdot \mathbf{C}[P,t]$. We denote this firing by $\mathbf{m} \xrightarrow{\alpha t} \mathbf{m}'$.*

Let us illustrate the dynamics of a CPN by the example depicted Figure 1. $p_1 + p_5 \xrightarrow{t_1} p_3 + p_5 \xrightarrow{\frac{1}{2}t_4} \frac{1}{2}p_1 + \frac{1}{2}p_2 + \frac{1}{2}p_3 + \frac{1}{2}p_5 \xrightarrow{\frac{1}{2}t_2} \frac{1}{2}p_1 + \frac{1}{2}p_4 + \frac{1}{2}p_5 \xrightarrow{\frac{1}{2}t_3} \frac{1}{2}p_1 + \frac{1}{2}p_3 + \frac{1}{2}p_5$ is a possible firing sequence. This interleaving semantic is appropriate when viewing CPNs as a relaxation of Petri nets. However it is interesting from a continuous point of view to extend it with a notion of limit (see next paragraph).

2.2. Properties of a CPN: specification and previous results

As in PNs, there are two kinds of properties: structural and behavioural ones. One goal of the net analysis is to relate behavioural properties to structural ones. Our work shows that this connexion is closer in CPNs than in PNs. Let us recall first basic structural properties of a CPN.

Definition 3 (Structural properties) *Let $\mathcal{N}$ be a CPN, then:*

- *Let* $\mathbf{x}$ *be a vector over places (resp. over transitions) then* $\mathbf{x}$ *is a P-semiflow (resp. T-semiflow) if* $\mathbf{x} \cdot \mathbf{C} = \mathbf{0}$ *(resp.* $\mathbf{C} \cdot \mathbf{x} = \mathbf{0}$*).*
- *$\mathcal{N}$ is said to be* conservative *(resp.* consistent*) if there exists* $\mathbf{x}$ *a P-semiflow (resp. T-semiflow) such that* $\mathbf{x} > \mathbf{0}$ *where the inegality is defined pointwise (i.e. $\forall a$ $\mathbf{x}[a] > 0$).*
- *A set of places Θ is a* trap *(resp. a* siphon*) iff $\Theta^\bullet \subseteq {}^\bullet\Theta$ (resp. ${}^\bullet\Theta \subseteq \Theta^\bullet$).*

In continuous PNs the reachability concept is not so immediate as in discrete nets. For example, in a continuous net it may happen that the marking of a place can be done smaller and smaller, but never reaches 0. This idea of getting as close as desired to a marking, even if it is never reached with a finite firing sequence leads in [12] to the definition of (pointwise) limit reachability, further refined in [10].

[b] Observe that the enabling degree may be infinite if t has no input places.

Definition 4 *Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a continuous system. We define two reachability sets:*

- $\mathrm{RS}(\mathcal{N}, \mathbf{m_0}) = \{\ \mathbf{m} \in (\mathbb{R}_{\geq 0})^{|P|} \mid$ *a finite fireable sequence* $\sigma = \alpha_1 t_{a_1} \dots \alpha_k t_{a_k}$ *exists such that* $\mathbf{m_0} \xrightarrow{\alpha_1 t_{a_1}} \mathbf{m}_1 \xrightarrow{\alpha_2 t_{a_2}} \dots \xrightarrow{\alpha_k t_{a_k}} \mathbf{m}_k = \mathbf{m}$ *with* $t_i \in T$ *and* $\alpha_i \in \mathbb{R}_{\geq 0}\}$.
- $\text{lim-RS}(\mathcal{N}, \mathbf{m_0}) = \{\mathbf{m} \in (\mathbb{R}_{\geq 0})^{|P|} |$ *a sequence of reachable markings* $\{\mathbf{m}_i\}_{i \geq 1}$ *exists verifying* $\mathbf{m_0} \xrightarrow{\sigma_1} \mathbf{m}_1 \xrightarrow{\sigma_2} \mathbf{m}_2 \cdots \mathbf{m}_{i-1} \xrightarrow{\sigma_i} \mathbf{m}_i \cdots$ *and* $\lim_{i \to \infty} \mathbf{m}_i = \mathbf{m}\}$.

The set of reachable (and lim-reachable) markings in continuous PNs satisfies some properties that do not hold for discrete nets. For example, the reachability set (and the lim-reachability set) of a continuous system is a convex set [12]. Many basic properties of discrete PNs can be extended to continuous PNs and also adapted to take into account the lim-reachability concept.

Definition 5 *Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a continuous system.*

- $\langle \mathcal{N}, \mathbf{m_0} \rangle$ *is* (lim-) *deadlock-free iff for every* $\mathbf{m} \in$ (lim-) $\mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ *there exists* $t \in T$ *such that* $\mathrm{enab}(t, \mathbf{m}) > 0$.
- $\langle \mathcal{N}, \mathbf{m_0} \rangle$ *is* (lim-) *live iff for every* $\mathbf{m} \in$ (lim-) $\mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ *and for every* $t \in T$ *there exist* $\mathbf{m}' \in$ (lim-) $\mathrm{RS}(\mathcal{N}, \mathbf{m})$ *such that* $\mathrm{enab}(t, \mathbf{m}') > 0$.
- $\langle \mathcal{N}, \mathbf{m_0} \rangle$ *is* (lim-) *reversible iff for every* $\mathbf{m} \in$ (lim-) $\mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ *then* $\mathbf{m_0} \in$ (lim-) $\mathrm{RS}(\mathcal{N}, \mathbf{m})$.

The support of a vector $\mathbf{v} \geq \mathbf{0}$ will be denoted as $\|\mathbf{v}\|$ and represents the set of positive elements of $\mathbf{v}$

Besides the situation in which the properties of the discrete and the continuous net are not related, it also happens that some properties of discrete PN cannot be observed in continuous systems, as mutex relationships. Moreover, the distinction between two properties may be lost in the continuous model. For example, under broad conditions, lim-liveness and lim-reversibility are equivalent. Indeed, the following result is a straightforward reformulation of Theorem 21 in [10].

Theorem 1 $\langle \mathcal{N}, \mathbf{m_0} \rangle$ *is consistent and* lim*-live iff it is* lim*-reversible and every transition is fireable at least once.*

However, liveness and consistency are not sufficient conditions for reversibility in discrete systems, and neither are for continuous net systems if finite firing sequences are considered. For example, the system in Fig. 1 is consistent and live as discrete, however once t_1 has fired it is impossible to get back to the initial marking. In the continuous net system, to go back to the initial marking from $\mathbf{m} = [0, 0, 1, 0, 1]$, an infinite sequence $\frac{1}{2}t_4 \frac{1}{2}t_2, \frac{1}{2}t_3, \frac{1}{4}t_4, \frac{1}{4}t_2, \frac{1}{4}t_3, \dots \frac{1}{2^k}t_4, \frac{1}{2^k}t_2, \frac{1}{2^k}t_3, \dots$ has to be fired.

The idea under continuization is that it leads to "easier to analyze" models. For that, we need to ensure that properties can be analyzed at least.

In [10], a characterization of reachability and lim-reachability is presented.

Theorem 2 *Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a CPN system and $\mathbf{m}$ be a marking.* $\mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0})$ *iff* $\exists \boldsymbol{\sigma} \in (\mathbb{R}_{\geq 0})^{|T|}$ *such that:*

1. $\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$

2. $\|\boldsymbol{\sigma}\| \in FS(\mathcal{N}, \mathbf{m_0})$

3. no trap in $P \setminus \|\mathbf{m}\|$ *intersects with* $\|\mathbf{m_0}\| \cup \|\boldsymbol{\sigma}\|^\bullet$

where $FS(\mathcal{N}, \mathbf{m_0})$ *is the set of the supports of sequences fireable from* $\mathbf{m_0}$*, which is a finite set that can be effectively constructed.*

A marking $\mathbf{m} \in$ lim-RS$(\mathcal{N}, \mathbf{m_0})$ *iff it verifies conditions (1) and (2).*

For the computation of $FS(\mathcal{N}, \mathbf{m_0})$, first add all the combinations of transitions that are enabled at $\mathbf{m_0}$. Then, take one of these sets and fire all the transitions, but in an amount smaller than the enabling degree. This will possibly enable other transitions, so new sets are added to FS. Repeat the procedure until all the sets in FS have been checked.

The only difference between reachability and lim-reachability is on traps, which can be emptied in the limit, but not with a finite sequence. In [10], decidability of reachability is proved. This result can be generalized to lim-reachability.

Corollary 1 *Reachability and* lim*-reachability are decidable for CPN systems.*

2.3. Analysis

We now show that the characterization of (lim-)reachability allows to address the (lim-)deadlock freeness and (lim-)liveness problems.

Theorem 3 *For CPN systems deadlock-freeness and* lim*-deadlock-freeness are decidable and belong to co-NP.*

Proof. Let us see that checking deadlock-freeness can be reduced to solving a set of linear programming problems.

Let $DP = \{DP_i\}_{i \in I}$ be the set of all the sets of places that have at least one input place per transition. Hence applying Theorem 2, $\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$ is a deadlock iff there exist $DP_i \in DP$ and $FS_j \in FS$ such that

$$\mathbf{m_0}[DP_i] + \mathbf{C}[DP_i, FS_j] \cdot \boldsymbol{\sigma}[FS_j] = \mathbf{0} \tag{1}$$

$$\mathbf{m_0}[P \setminus DP_i] + \mathbf{C}[P \setminus DP_i, FS_j] \cdot \boldsymbol{\sigma}[FS_j] > \mathbf{0} \tag{2}$$

$$\boldsymbol{\sigma}[FS_j] > \mathbf{0} \tag{3}$$

$$\text{For every trap } \Theta \subseteq DP_i, (\|\mathbf{m_0}\| \cup FS_j{}^\bullet) \cap \Theta = \emptyset \tag{4}$$

Applying the characterization of traps in [15], (4) is equivalent to checking whether the solution of the following linear programming problem is zero,

$$\begin{aligned} &\text{maximize } \varepsilon_{i,j} \\ &\text{subject to: } \mathbf{y} \cdot \mathbf{C}_\Theta \geq \mathbf{0} \\ &\qquad \mathbf{y} \geq \mathbf{0} \\ &\qquad \mathbf{y}[P \setminus DP_i] = \mathbf{0} \\ &\qquad \sum_{p \in \|\mathbf{m_0}\| \cup FS_j{}^\bullet} \mathbf{y}[p] \geq \varepsilon_{i,j} \end{aligned} \tag{5}$$

where C_Θ is the token flow matrix of the net $\mathcal{N}_\Theta = \langle P, T, \mathbf{Pre}, \mathbf{Post}_\Theta \rangle$ with $\mathbf{Post}_\Theta[p,t] = 0$ iff $\mathbf{Post}[p,t] = 0$, and $\mathbf{Post}_\Theta[p,t] \geq \sum_{p' \in {}^\bullet t} \mathbf{Pre}[p',t]$.

Equations (1), (2) and (3) are equivalent to:

$$\begin{aligned} &\text{maximize } \delta_{i,j} \\ &\text{subject to: } \mathbf{m_0}[DP_i] + \mathbf{C}[DP_i, FS_j] \cdot \boldsymbol{\sigma}[FS_j] = \mathbf{0} \\ &\qquad \mathbf{m_0}[P \setminus DP_i] + \mathbf{C}[P \setminus DP_i, FS_j] \cdot \boldsymbol{\sigma}[FS_j] \geq \delta_{i,j} \cdot \mathbf{1} \\ &\qquad \boldsymbol{\sigma}[FS_j] \geq \delta_{i,j} \cdot \mathbf{1} \end{aligned} \tag{6}$$

Therefore, to prove that the system has a deadlock, check that there exist $DP_i \in DP$ and $FS_j \in FS$ such that (5) has no positive solution and (6) has a positive solution. The guesses and the linear problem computations can be done in polynomial time. So this leads to the complexity stated in the theorem.

Regarding lim-deadlock-freeness, the only difference is that there is no restriction with respect to the traps, so clearly it is also decidable with the same complexity. □

Theorem 4 *Liveness and* lim*-liveness are decidable for CPN systems and belong to co-NP.*

Proof. Let us study liveness first. For continuous PNs, a transition t is not fireable for any successor of $\mathbf{m}$ iff there exists an empty siphon in $\mathbf{m}$ that contains a place $p \in {}^\bullet t$ (Lemma 11 in [10]). Hence, the system is non live iff there exist $FS_j \in FS$ and a siphon Σ such that

$$\mathbf{m} = \mathbf{m_0} + \mathbf{C}[P, FS_j] \cdot \boldsymbol{\sigma}[FS_j] \tag{7}$$

$$\mathbf{m} \geq \mathbf{0} \tag{8}$$

$$\boldsymbol{\sigma}[FS_j] > \mathbf{0} \tag{9}$$

$$\mathbf{m}[\Sigma] = 0 \tag{10}$$

$$\text{For any trap } \Theta \text{ such that } (\|\mathbf{m_0}\| \cup FS_j{}^\bullet) \cap \Theta = \emptyset, \exists p \in \Theta \text{ with } \mathbf{m}[p] = 0 \tag{11}$$

Let $\{\Sigma_i\}_{i \in I}$ be the set of minimal siphons (i.e., siphons that are not contained in other siphons), and let $\{\Theta_k\}_{k \in K}$ be the set of traps. Then for each $FS_j \in FS$, each trap Θ_k verifying $(\|\mathbf{m_0}\| \cup FS_j{}^\bullet) \cap \Theta_k \neq \emptyset$, and each siphon Σ_i, check whether the following linear programming problem has a positive solution:

$$\begin{aligned} &\text{maximize } \varepsilon_{i,j} \\ &\text{subject to: } \mathbf{m} = \mathbf{m_0} + \mathbf{C}[P, FS_j] \cdot \boldsymbol{\sigma}[FS_j] \\ &\qquad \mathbf{m} \geq \mathbf{0} \\ &\qquad \mathbf{m}[\Sigma_i] = 0 \\ &\qquad \boldsymbol{\sigma}[FS_j] \geq \varepsilon_{i,j} \cdot \mathbf{1} \\ &\qquad \mathbf{m}[p] = 0 \end{aligned}$$

Since solving linear programming is done in polynomial time, the stated complexity bound is obtained. Regarding lim-liveness, notice that in fact it is equivalent that a transition is not fireable from any reachable marking or from any lim-reachable marking. Then, the lim-liveness problem is analogous, changing the reachability

condition, which amounts to forgetting the traps, and removing the last equation ($\mathbf{m}[p] = 0$) in each one of the linear programming problems. □

The deadlock-freeness and the liveness problems are EXPSPACE-hard for Petri nets and PSPACE-complete for safe Petri nets. Thus, as claimed above, problems in CPNs can be solved more efficiently.

3. Timed Continuous Petri Nets

3.1. Syntax and Semantics

A simple and interesting way to introduce time in discrete PNs is to assume that all the transitions are timed with exponential probability distribution functions. For the timing interpretation of continuous PNs we will use a first order (or deterministic) approximation of the discrete case, assuming that the delays associated to the firing of transitions can be approximated by their mean values. These mean delays will be assumed to be positive, i.e., immediate transitions are not allowed.

Definition 6 *A* Timed Continuous Petri Net (TCPN) *is a continuous PN s.t.* $\forall t \in T, |{}^\bullet t| \geq 1$ *together with a vector* $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^{|T|}$.
A TCPN with an initial marking $\langle \mathcal{N}, \boldsymbol{\lambda}, \mathbf{m_0} \rangle$ *will be denoted a TCPN system.*

Since it is an interpretation of the autonomous net, the evolution of a TCPN has to fulfill the state equation: $\mathbf{m}(\tau) = \mathbf{m}(0) + \mathbf{C} \cdot \boldsymbol{\sigma}(\tau)$, where $\mathbf{m}$ and σ now depend on τ, the actual time. Deriving, $\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \dot{\boldsymbol{\sigma}}(\tau)$ (with $\dot{\mathbf{m}}(\tau) = \frac{d\mathbf{m}}{d\tau}(\tau)$). $\dot{\boldsymbol{\sigma}}(\tau)$, the flow obtained by firing the transitions, depends only on the current marking and thus can be rewritten as $\dot{\boldsymbol{\sigma}}(\tau) = \mathbf{f}(\mathbf{m}(\tau))$. Different semantics have been used to define the flow function $\mathbf{f}$, the two most important being *infinite server* (or *variable speed*) and *finite server* (or *constant speed*) [1, 14]. Here infinite server semantics will be considered.

Like in purely markovian discrete net models, under *infinite server semantics*, the flow through a timed transition t is the product of the speed, $\boldsymbol{\lambda}[t]$, and $\text{enab}(t, \mathbf{m})$, the instantaneous enabling of the transition, i.e.,

$$\mathbf{f}(\mathbf{m})[t] = \boldsymbol{\lambda}[t] \cdot \text{enab}(t, \mathbf{m}) = \boldsymbol{\lambda}[t] \cdot \min_{p \in {}^\bullet t} \{\mathbf{m}[p] / \mathbf{Pre}[p, t]\}.$$

For the flow to be well defined, every transition must have at least one input place, hence the restriction included in the definition of TCPNs. To summarize, the differential equations ruling the behaviour of a TCPNs are:

$$\forall p \in P \ \dot{\mathbf{m}}[p] = \sum_{t \in T} C[p, t] \cdot \boldsymbol{\lambda}[t] \cdot \min_{p' \in {}^\bullet t} \{\mathbf{m}[p'] / \mathbf{Pre}[p', t]\}$$

Notice that the flow is a piecewise linear function. The change of behavior happens when in a synchronization the place representing the minimum changes. Hence, the switching among the linear systems is given by an internal event. A system without synchronizations (i.e., for every t $|{}^\bullet t| = 1$) would be linear.

Let us introduce the concept of *configurations*: a configuration assigns to a transition one place that for some markings will control its firing rate. Thus the

number of configurations is at most $\Pi_{t\in T}|^\bullet t|$. The reachability space can be divided into *regions* according to the configurations. These regions are polyhedrons, and are disjoint, except on the borders. Inside each polyhedron, the evolution of the system is defined by a linear differential equation.

3.2. Timed Differentiable Petri nets

The structure and equations of TCPN were somehow inherited from those of discrete Petri nets. Petri nets are based on a production/consumption logic, and it is the flow of material that is mainly represented. However, these material flow channels can also be used to simulate control flows, and self-loop structures appear. This kind of structures can be used in continuous PNs to "force" behaviors.

Moreover, the information that appears in the structure of the net is redundant. Assume a place-transition-place-transition subnet, and let us denote them as $p_1 - t_1 - p_2 - t_2$. The marking evolution of p_2 due to the output flow does not depend on the weight of the arc (p_2, t_2), and its input flow really depends on the quotient of the weights (p_1, t_1) and (t_1, p_2).

Hence, the notation of TCPN "looks cumbersome", and the evolution rules seem convoluted and counterintuitive in some cases. Trying to clarify the behavior, in [6], a different way of introducing time in a PN structure was presented, in which two different kinds of arcs are used: one to model the *control*, and the other to model the *marking evolution*. This is similar to what is done in Forrester diagrams, in which *control* and *material* flows are kept separate [5].

Definition 7 *A Timed Differentiable Petri Net (TDPN)* $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W}\rangle$ *is defined by:*

- $\langle P, T, \mathbf{C}\rangle$*, a pure PN (thus* $\mathbf{C}$ *is the incidence matrix),*
- $\mathbf{W}$*, the speed control matrix: a mapping from* $P \times T$ *to* $\mathbb{R}_{\geq 0}$ *such that:* $\forall t \in T, \exists p \in P, \mathbf{W}(p,t) > 0$ *and* $\forall t \in T, \forall p \in P, \mathbf{C}(p,t) < 0 \Rightarrow \mathbf{W}(p,t) > 0$

A TDPN with an initial marking $\langle \mathcal{D}, \mathbf{m_0}\rangle$ *will be denoted a TDPN system.*

The first requirement about $\mathbf{W}$ ensures that the firing rate of any transition is defined, whereas the second one ensures that the marking remains non negative. Notice that these weights are defined as real numbers, while in TCPN all the arc weights are natural numbers.

We are now in position to give semantics to TDPNs.

Definition 8 *Let* $\mathcal{D}$ *be a TDPN. A trajectory is a continuously differentiable mapping* $\mathbf{m}$ *from time (i.e.,* $\mathbb{R}_{\geq 0}$*) to the set of markings (i.e.,* $(\mathbb{R}_{\geq 0})^P$*) which satisfies the following differential equation system:*

$$\begin{aligned} \dot{\mathbf{m}} &= \mathbf{C} \cdot \mathbf{f}(\mathbf{m}) \\ \mathbf{f}(\mathbf{m})[t] &= \min(\mathbf{W}(p,t) \cdot \mathbf{m}[p] \mid \mathbf{W}(p,t) > 0) \end{aligned} \tag{12}$$

Graphical notations. We extend the graphical notations of PN to take into account $\mathbf{W}$. These arcs are not oriented, since they are always defined as preconditions of transitions. Like in Forrester diagrams, to help distinguishing the $\mathbf{W}$ arcs

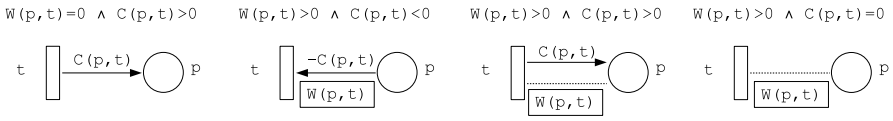

Figure 2: Graphical notations

from the **Pre** and **Post** arcs, **W** arcs will be drawn with dotted lines. To distinguish between the labels, $\mathbf{W}(p,t)$ will be drawn inside a box. There are four possible patterns illustrated in Fig. 2. When $\mathbf{W}(p,t) = 0 \wedge \mathbf{C}(p,t) > 0$, place p receives tokens from t and does not control its firing rate. When $\mathbf{W}(p,t) > 0 \wedge \mathbf{C}(p,t) < 0$, place p provides tokens to t. So it *must control* its firing rate. Hence, the non oriented arc between p and t is redundant, and we will not draw it and represent only an oriented arc from p to t both labelled by $-\mathbf{C}(p,t)$ and $\mathbf{W}(p,t)$. When $\mathbf{W}(p,t) > 0 \wedge \mathbf{C}(p,t) > 0$, place p receives tokens from t and controls its firing rate. There is both an oriented arc from t to p and a non oriented arc between p and t with their corresponding labels. When $\mathbf{W}(p,t) > 0 \wedge \mathbf{C}(p,t) = 0$, place p controls the firing rate of t and t does not modify the marking of p, so there is a non oriented arc between p and t. As usual, we omit labels when equal to 1.

As an example, the equations in (13) correspond to the TDPN in Fig. 3.

$$\begin{aligned}
\dot{\mathbf{m}}[x_1] &= 2a\omega \cdot \min\left\{\frac{\mathbf{m}[x_2]}{2a}, \mathbf{m}[y_1]\right\} - a\omega \cdot \min\left\{\mathbf{m}[x_1], \frac{\mathbf{m}[pk]}{a}\right\} \\
\dot{\mathbf{m}}[x_2] &= a\omega \cdot \min\left\{\mathbf{m}[y_2], \frac{\mathbf{m}[pk]}{a}\right\} - 2a\omega \cdot \min\left\{\mathbf{m}[x_2], \frac{\mathbf{m}[x_1]}{2a}\right\} \\
\dot{\mathbf{m}}[y_1] &= a\omega \cdot \min\left\{\mathbf{m}[x_1], \frac{\mathbf{m}[pk]}{a}\right\} - 2a\omega \cdot \min\left\{\frac{\mathbf{m}[x_2]}{2a}, \mathbf{m}[y_1]\right\} \\
\dot{\mathbf{m}}[y_2] &= 2a\omega \cdot \min\left\{\mathbf{m}[x_2], \frac{\mathbf{m}[x_1]}{2a}\right\} - a\omega \cdot \min\left\{\mathbf{m}[y_2], \frac{\mathbf{m}[pk]}{a}\right\}
\end{aligned} \tag{13}$$

This net has sixteen configurations. However, for $1 \leq a \leq b \leq 2a-1$, the system never switches and always remains in the configuration defined by $\mathbf{m_0}$.

The Petri net structure of TCPN is similar to that of TDPN. Moreover, the minimum operator appears related to the marking evolution. Hence, it would not be surprising that both models are equivalent, as long as the elements of the speed control matrix are in the rationals. To prove that, observe first that fractions in the **Pre** and **Post** matrices do not pose a problem, since they can be easily avoided multiplying the columns of the **Pre** and **Post** matrices by the right number (this does not change the dynamics of the system).

Proposition 1 *For any TCPN, a TDPN with the same number of places and transitions exists which has the timed evolution, and vice versa (as long as the weights of the TDPN are rational numbers).*

Proof. Let $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W}\rangle$ be a TDPN, and let us construct a TCPN $\mathcal{S} = \langle P, T, \mathbf{Pre}', \mathbf{Post}', \boldsymbol{\lambda}\rangle$ with the same differential equations. For each $t \in T$, define $\mathbf{W}^*(t) \geq \max(-\mathbf{C}(p,t) \cdot \mathbf{W}(p,t) \mid \mathbf{C}(p,t) < 0)$ (i.e., $\mathbf{W}^*(t)$ can be defined as any

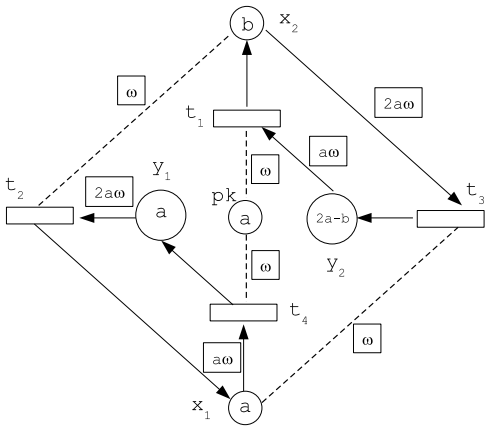

Figure 3: A periodic TDPN

value that fulfills this inequality). By definition it is greater than zero. For each $p \in P$ and $t \in T$, define

- $\mathbf{Pre}'(p,t) = \mathbf{W}^*(t)/\mathbf{W}(p,t)$ if $\mathbf{W}(p,t) \neq 0$, and $\mathbf{Pre}'(p,t) = 0$ otherwise.
- $\mathbf{Post}'(p,t) = \mathbf{C}(p,t) + \mathbf{W}^*(t)/\mathbf{W}(p,t)$ if $\mathbf{W}(p,t) \neq 0$, and $\mathbf{Post}'(p,t) = \mathbf{C}(p,t)$ otherwise.
- $\boldsymbol{\lambda}(t) = \mathbf{W}^*(t)$

Let now $\mathcal{S} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \boldsymbol{\lambda} \rangle$ be a TCPN, and let us construct a TDPN $\mathcal{D} = \langle P, T, \mathbf{C}', \mathbf{W} \rangle$ with the same differential equations. For each $p \in P$ and $t \in T$, define

- $\mathbf{C}'(p,t) = \mathbf{Post}(p,t) - \mathbf{Pre}(p,t)$
- $\mathbf{W}(p,t) = \boldsymbol{\lambda}(t)/\mathbf{Pre}(p,t)$ if $p \in {}^\bullet t$, and 0 otherwise.

□

The patterns appearing on the right of Fig. 2 represent situations in which a place acts as a control place of a transition for which it is not an input place (in the sense that the transition is not removing tokens from the place). That is, they represent non-consuming control arcs. However, these patterns can be simulated with the two ones on the left.

Proposition 2 *Let $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W} \rangle$ be a TDPN. Then another TDPN $\mathcal{D}' = \langle P', T', \mathbf{C}', \mathbf{W}' \rangle$ can be defined, using only the two patterns on the left of Fig. 2, with the same timed evolution as $\mathcal{D}$ but for some duplicated places. Moreover, the transformation is linear in size, since it at most doubles the number of places and transitions.*

Proof. Define the new TDPN as follows:

- $P' = \{p^-, p^+ \mid p \in P\}$

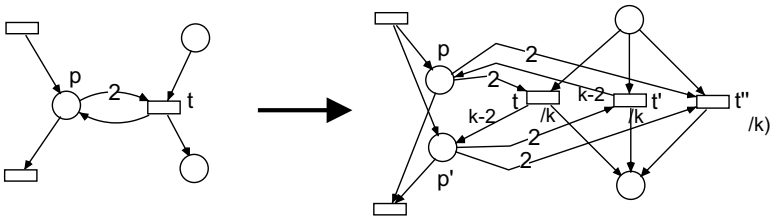

Figure 4: TCPN can always be transformed into pure nets.

- $T' = \{t^-, t^+ \mid t \in T\}$
- To define $\mathbf{C}'$, distinguish two cases:
  - If $\mathbf{C}(p,t) < 0 \vee \mathbf{W}(p,t) = 0$ then
    $\mathbf{C}'(p^-,t^-) = \mathbf{C}'(p^+,t^+) = \mathbf{C}(p,t)$ and $\mathbf{C}'(p^-,t^+) = \mathbf{C}'(p^+,t^-) = 0$
  - If $\mathbf{C}(p,t) \geq 0 \wedge \mathbf{W}(p,t) > 0$ then
    $\mathbf{C}'(p^-,t^-) = \mathbf{C}'(p^+,t^+) = -1$ and $\mathbf{C}'(p^-,t^+) = \mathbf{C}'(p^+,t^-) = \mathbf{C}(p,t)+1$
- $\mathbf{W}'(p^-,t^-) = \mathbf{W}'(p^+,t^+) = \mathbf{W}(p,t)$ and $\mathbf{W}'(p^-,t^+) = \mathbf{W}'(p^+,t^-) = 0$

□

The transformation that has been proposed doubles the number of places and transitions. In practice, sometimes it is not necessary to duplicate all of them. A similar transformation can also be applied to *remove self-loop arcs* in TCPN. Let us see the idea in the case of one self-loop (see Fig. 4). The transformation replaces the self-loop with two places and three transitions. These two places will have the same marking the original place had, and the flow of the original transition is now split into the flow of these three transitions. Any other input/output place of the transition is input/output of all the transitions. Any input/output transition of the place is now input/output of all the places. The rates of these new transitions are defined so that the sum of their flows is equal to the flow of the original transition.

Again, this is a linear transformation. The procedure can be easily generalized to the case in which several places are connected with self-loops to the same transition (just duplicate each self-loop place and split the transition in three), or when one place is engaged in several self-loops (duplicate the self-loop place and split each transition in three).

Proposition 3 *For any TCPN a pure TCPN can be defined which has the same timed evolution, but for some duplicated places. Moreover, the transformation is linear in size, since it at most doubles the number of places and triples the number of transitions.*

The results have been summarized in the schema in Fig. 5.

Theorem 5 *The expressive power of TDPN, TDPN constrained to the use of the two basic constructions (the two on the left in Fig. 2), TCPN and pure TCPN are identical. Moreover, the transformations range from keeping the places and transitions to a linear increase in size (see the diagram in Fig. 5).*

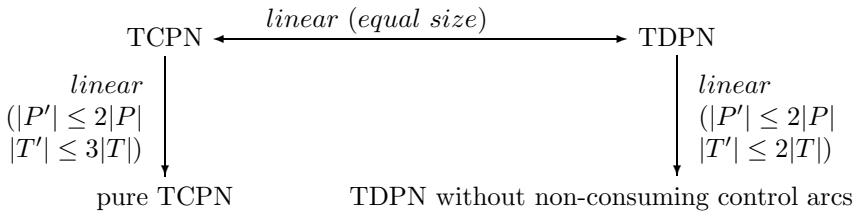

Figure 5: Relationships between TDPN and TCPN and their versions without self-loops or non-consuming control arcs

4. Undecidablity results for TDPNs

In this section, we study the expressiveness of TDPNs (equivalent to TCPNs as shown in the previous section). More precisely we want to prove that TDPNs are at least as expressive as Turing machines or any equivalent mechanism. However such a result requires to formally define how a continuous (w.r.t. time and space) system simulates a discrete one.

4.1. Dynamical systems and simulation

Definition 9 *A deterministic dynamical system $(X, \mathcal{T}, f)$ is defined by:*
- *a state space X, a time space $\mathcal{T}$ ($\mathcal{T}$ is either $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$),*
- *a transition function f from $X \times \mathcal{T}$ to X fulfilling:*

$$\forall x \in X, \forall \tau_1, \tau_2 \in \mathcal{T}, f(x, 0) = x \wedge f(x, \tau_1 + \tau_2) = f(f(x, \tau_1), \tau_2)$$

In the sequel, we will only deal with deterministic systems. In a *discrete* (resp. *continous*) system $X \subseteq \mathbb{N}^d$ for some d (resp. $X \subseteq (\mathbb{R}_{\geq 0})^d$) and $\mathcal{T} = \mathbb{N}$ (resp. $\mathcal{T} = \mathbb{R}_{\geq 0}$). The simulation of a discrete system by a continuous one involves a mapping from the set of states of the discrete system to the powerset of states of the continuous systems and an observation epoch. A simulation ensures that, starting from some state in the image of an initial state of the discrete system and observing the state reached after some multiple n of the epoch, one can recover the state of the discrete system after n steps. If the continuous system evolves in some bounded subset of $(\mathbb{R}_{\geq 0})^d$, the simulation is said *bounded*.

Definition 10 *A continuous dynamical system $(Y, \mathbb{R}_{\geq 0}, g)$ simulates a discrete dynamical system $(X, \mathbb{N}, f)$ if there is a mapping ϕ from X to 2^Y and $\tau_0 \in \mathbb{R}_{>0}$ such that:*
- $\forall x \neq x' \in X, \phi(x) \cap \phi(x') = \emptyset$
- $\forall x \in X, \forall y \in \phi(x), g(y, \tau_0) \in \phi(f(x, 1))$

The simulation is said bounded *if $Y \subset [0, K]^d$ for some $K \in \mathbb{R}_{\geq 0}$.*

Roughly speaking, a *robust* simulation is insensitive to small perturbations of the simulation mapping and the observation instants. In order to define robust simulation, we refine the notion of simulation. First, a *two-level* dynamical system $(Y = Y_1 \times Y_2, \mathbb{R}_{\geq 0}, g)$ is such that g is defined by g_1 from $Y_1 \times \mathbb{R}_{\geq 0}$ to Y_1 and by g_2 from $Y \times \mathbb{R}_{\geq 0}$ to Y_2 as: $g((y_1, y_2), \tau) = (g_1(y_1, \tau), g_2((y_1, y_2), \tau))$. In words, the behaviour of the first component depends only on its local state.

Definition 11 *A two-level continuous dynamical system* $(Y, \mathbb{R}_{\geq 0}, g)$ consistently *simulates a discrete dynamical system* $(X, \mathbb{N}, f)$ *if there is* $y_0 \in Y_1$*, a mapping* ϕ *from* X *to* 2^{Y_2} *and* $\tau_0 \in \mathbb{R}_{>0}$ *such that:*

- $\forall x \neq x' \in X, \phi(x) \cap \phi(x') = \emptyset$,
- $g_1(y_0, \tau_0) = y_0$,
- $\forall x \in X, \forall y \in \phi(x), g_2((y_0, y), \tau_0) \in \phi(f(x, 1))$.

Note that the first part of component is a "fixed" part of the system since its whole trajectory does not depend on the input of the simulated system.

Definition 12 *A simulation (by a two-level system) is* robust *iff there exists* $\delta, \epsilon \in \mathbb{R}_{>0}$ *such that:*

- $\forall x \neq x' \in X, dist(\phi(x), \phi(x')) > 2\epsilon$
- $\forall x \in X, \forall y_2 \in Y_2, \forall n \in \mathbb{N}, \forall \tau \in \mathbb{R}_{\geq 0}$,
  $max(dist(y_2, \phi(x)), dist(\tau, n\tau_0)) \leq \delta \Rightarrow dist(g_2((y_0, y_2), \tau), \phi(f(y, n))) \leq \epsilon$

where $dist(Y, Y') = \inf(|y - y'|_\infty \mid y \in Y, y' \in Y')$

Thus, if the simulation is robust, starting with an initial state no more pertubated than δ and delaying or anticipating the observation of the system by no more than δ, the state of the simulated system can be recovered. For obvious reasons, the simulation of an infinite-state system cannot be *simultaneously robust and bounded*.

4.2. Two counter machines

We will simulate two (non negative integer) counter machines (equivalent to Turing machines [9]). Their behaviour is described by a set of instructions. An instruction `I` may be one of the following kind with an obvious meaning ($\texttt{cpt}_\texttt{u}$ is a counter with $u \in \{1, 2\}$):

- `I : goto I'`;
- `I : increment`($\texttt{cpt}_\texttt{u}$); `goto I'`;
- `I : decrement`($\texttt{cpt}_\texttt{u}$); `goto I'`;
- `I : if` $\texttt{cpt}_\texttt{u} = 0$ `then goto I' else goto I`";
- `I : STOP`;

W.l.o.g. a decrementation must be preceded by a test on the counter and the (possible) successor(s) of an instruction is (are) always different from it.

4.3. Basic principles of the simulation

4.3.1. Transition pairs.

In a TDPN, when a transition begins to fire, it will never stop. Thus we use *transition pairs* in order to *temporarily* either move tokens from one place to another one, or produce/consume tokens in a place.

Let us examine transitions t_{high} and t_{low} of figure 6. Their incidence is opposite. So if their firing rate is equal no marking change will occur. Let us examine $\mathbf{W}$, all the items of $\mathbf{W}(t_{high})$ and $\mathbf{W}(t_{low})$ are equal except $\mathbf{W}(pk, t_{low}) = k$ and $\mathbf{W}(pk, t_{high}) = \perp$. Thus, if any other place controls the firing rate of t_{low} it will be equal to the one of t_{high}. Place pk is a *constant* place meaning that its marking

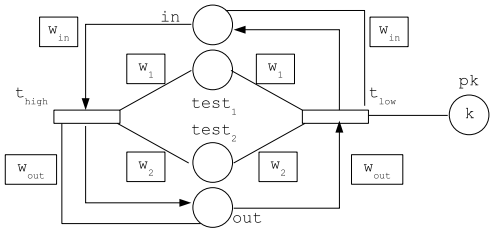

Figure 6: A transition pair

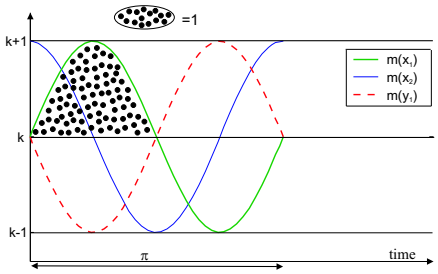

Figure 7: The behaviour of the clock subnet

will always be k. Summarizing:

– if $w_{in}\mathbf{m}(in) > k \wedge w_{out}\mathbf{m}(out) > k \wedge w_1\mathbf{m}(test_1) > k \wedge w_2\mathbf{m}(test_2) > k$ then this pair transfers some amount of tokens from *in* to *out*,

– otherwise, there will be no marking change.

The clock subnet. The net that we build consists in two subnets: an instance of the subnet of figure 3, called in the sequel the *clock* subnet, and another subnet depending on the counter machine called the *operating* subnet. The clock subnet has k as average value, 1 as amplitude and π as period (i.e. $\omega = 2$). We recall the behavioural equations of the place markings that will be used by the operating subnet: $\mathbf{m}(x_1)(\tau) = k + sin(2\tau), \mathbf{m}(y_1)(\tau) = k - sin(2\tau)$.

Figure 7 represents the evolution of markings for x_1, y_1 and x_2 (the marking of place y_2 is symmetrical to x_2 w.r.t. the axis $\mathbf{m} = k$). Note that the mottled area is equal to 1.

The marking changes of the operating subnet will be ruled by the places x_1 and y_1. An *execution cycle* of the net will last π. The first part of the cycle (i.e. $[h\pi, h\pi + \pi/2]$ for some $h \in \mathbb{N}$) corresponds to $\mathbf{m}(x_1) \geq k$ and the second part of the cycle (i.e. $[h\pi + \pi/2, (h+1)\pi]$) corresponds to $\mathbf{m}(y_1) \geq k$. So, the period of observation τ_0 is equal to π.

Specialisation of the transition pairs pattern.

Using place x_1 (or y_1), we specialise transition pairs as illustrated in figure 8 (pk is the constant place of the clock subnet). In this subnet, one of the test place is x_1 and the control weights of the two test places (x_1 and *test*) are 1. First due to the periodical behaviour of $\mathbf{m}(x_1)$, no tokens transfer will occur during the second part

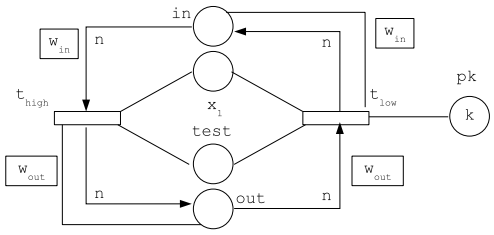

Figure 8: A specialised transition pair

of the cycle. Let us examine the different cases during a time interval $[h\pi, h\pi+\pi/2]$. We assume that within this interval $\mathbf{m}(test), \mathbf{m}(in)$ and $\mathbf{m}(out)$ are not modified by the other transitions.

- If $\mathbf{m}(test)(h\pi) \leq k$ then there will be no transfer of tokens.
- If $\mathbf{m}(test)(h\pi) \geq k+1 \wedge w_{in}(\mathbf{m}(in)(h\pi) - n) \geq k+1 \wedge w_{out}\mathbf{m}(out)(h\pi) \geq k+1$ then t_{high} will be controlled by x_1 and t_{low} will be controlled by pk. Hence (see the integral of figure 7) exactly n tokens will be transfered from in to out.
- Otherwise, some amount of tokens in $[0, n]$ will be transfered from in to out.

From a simulation point of view, one wants to avoid the last case. For the same reason, when possible, we choose w_{in} and w_{out} enough large so that it ensures that in and out will never control t_{high} and t_{low}.

4.4. The operating subnet

Places of the operating subnet and the simulation mapping. Let us suppose that the counter machine has l instructions $\{\mathtt{I}_1, \ldots, \mathtt{I}_l\}$ and two counters $\{\mathtt{cpt}_1, \mathtt{cpt}_2\}$. The operating subnet has the following places: pc, qc, pn, qn, c_1, c_2. The forth first places simulate the program counter whereas the last ones simulate the counters. Furthermore by construction, the following invariants will hold for every reachable marking $\mathbf{m}$: $\mathbf{m}(pc) + \mathbf{m}(qc) = l + 1$ and $\mathbf{m}(pn) + \mathbf{m}(qn) = l + 1$. We now define the simulation mapping ϕ. Assume that, in a state s of the counter machine, $\mathtt{I_i}$ is the next instruction and the value of the counter $\mathtt{cpt_u}$ is v_u. Then a marking $\mathbf{m} \in \phi(s)$ iff:

- The submarking corresponding to the clock subnet is its initial marking.
- $\mathbf{m}(pn) = i$, $\mathbf{m}(qn) = l + 1 - i$,
  if $1 < i < l$ then
    $\mathbf{m}(pc) \in [i - l/k, i + l/k]$ and $\mathbf{m}(qc) \in [l + 1 - i - l/k, l + 1 - i + l/k]$
  else if $i = 1$
    $\mathbf{m}(pc) \in [1, 1 + l/k]$ and $\mathbf{m}(qc) \in [l - l/k, l]$
  else if $i = l$
    $\mathbf{m}(pc) \in [l - l/k, l]$ and $\mathbf{m}(qc) \in [1, 1 + l/k]$
- $\mathbf{m}(c_1) = k - 1 + 3v_1, \mathbf{m}(c_2) = k - 1 + 3v_2$.

Moreover, we choose $k \geq 6l^2$ for technical reasons.

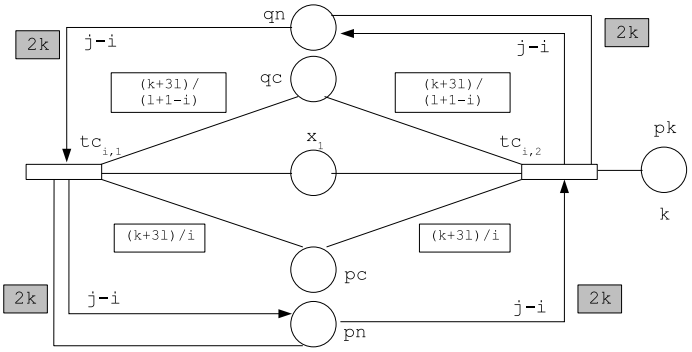

Figure 9: First stage: simulation of an unconditional jump

Principle of the instruction simulation. The simulation of an instruction $\mathtt{I_i}$ takes exactly the time of the cycle of the clock subnet and is decomposed in two parts ($\mathbf{m}(x_1) \geq k$ followed by $\mathbf{m}(y_1) \geq k$).

The first stage is triggered by $((k+3l)/i)\mathbf{m}(pc) \geq k+1 \wedge ((k+3l)/(l+1-i)\mathbf{m}(qc) \geq k+1$ and performs the following tasks:

- updating $\mathbf{m}(pn)$ by producing (resp. consuming) $j-i$ tokens if $j > i$ (resp. $j < i$) where $\mathtt{I_j}$ is the next instruction; simultaneously updating $\mathbf{m}(qn)$ accordingly. If $\mathtt{I_i}$ is a conditional jump, this involves to find the appropriate j. The marking of pn will vary from i to j and the one of qn from $l+1-i$ to $l+1-j$,
- updating the counters depending on the instruction.

The second stage is triggered by $((k+3l)/j)\mathbf{m}(pn) \geq k+1 \wedge ((k+3l)/(l+1-j)\mathbf{m}(qn) \geq k+1$ and performs the following task: updating $\mathbf{m}(pc)$ and $\mathbf{m}(qc)$ by a variable value in such a way that their marking still belong to the intervals associated with the simulation mapping.

First stage: simulation of an unconditional jump. This part of the simulation applies to both an unconditional jump, an incrementation and a decrementation. The simulation of the counter updates is straightforward once this pattern is presented. For this kind of instructions, the next instruction say $\mathtt{I_j}$ is *a priori* known.

The subnet we build depends on the relative values of i (the index of the current instruction) and j (the index of the next instruction). Here, we assume that $i < j$, the other case is similar. The transition pair of figure 9 is both triggered by pc and qc.

- Assume that the current instruction is $\mathtt{I_{i'}}$ with $i' \neq i$. If $i' < i$ then pc disables the transition pair whereas if $i' > i$ then qc disables the transition pair. We explain the first case. $\mathbf{m}(pc) \leq i' + l/k \leq i - 1 + l/k$; thus $((k+3l)/i)\mathbf{m}(pc) \leq ((k+3l)/i)(i-1+l/k) \leq k-1$ (due to our hypothesis on k).
- Assume that the current instruction is $\mathtt{I_i}$. Then both $((k+3l)/i)\mathbf{m}(pc) \geq ((k+3l)/i)(i-l/k) \geq k+2$ and

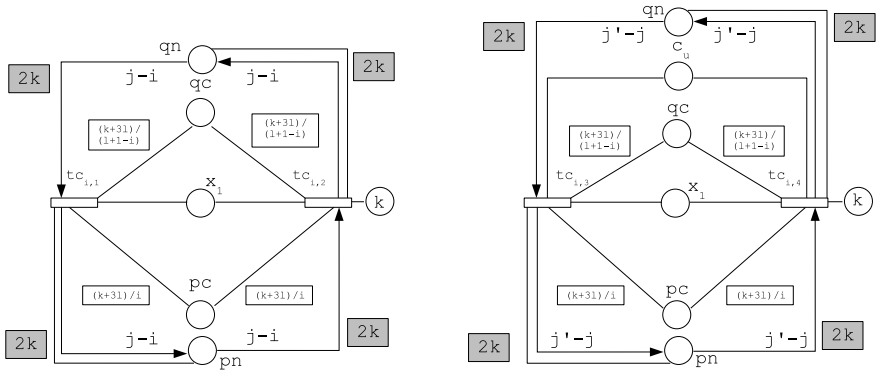

Figure 10: The first stage of a conditional jump (places are duplicated for readability)

$((k+3l)/(l+1-i))\mathbf{m}(qc) \geq ((k+3l)/(l+1-i))((l+1-i)-l/k) \geq k+2$.

Thus in the second case, the pair is activated and transfers $j-i$ tokens from qn to pn during the first part of the cycle as required.

Note that $\mathbf{W}(pn, tc_{i,1}) = \mathbf{W}(pn, tc_{i,2}) = \mathbf{W}(qn, tc_{i,1}) = \mathbf{W}(qn, tc_{i,2}) = 2k$ ensures that places pn and qn do not control these transitions (since $2k \geq k+2$ for k enough large).

First stage: simulation of a conditional jump. The first stage for simulating the instruction $\mathtt{I_i}$: `if` $\mathtt{cpt_u} = 0$ `then goto` $\mathtt{I_j}$ `else goto` $\mathtt{I_{j'}}$; is illustrated in figure 10 in case $i < j < j'$ (the other cases are similar).

It consists in two transition pairs. Pair $tc_{i,1}, tc_{i,2}$ mimics the first stage of an unconditional jump from $\mathtt{I_i}$ to $\mathtt{I_j}$. It will transfer during the first part of the cycle $j-i$ tokens from qn to pn. Pair $tc_{i,3}, tc_{i,4}$ is triggered if $c_u \geq k+1$ (i.e. the counter $\mathtt{cpt_u}$ is non null). If it is the case it will transfer $j'-j$ tokens from qn to pn. Thus:

- If $\mathbf{m}(c_u) = k-1$ then only the first pair is triggered and $j-i$ tokens will be transfered from qn to pn.
- otherwise $\mathbf{m}(c_u) \geq k+2$, the two pairs are simultaneously triggered and $j-i$ tokens will be transfered from qn to pn and $j'-j$ from qn to pn. Summing, $j'-i$ tokens will be transfered from qn to pn as required.

The second stage. This stage is the *difficult* part of this simulation. Due to the fact that the ODE ruling a TDPN is a linear equation inside a configuration, we cannot obtain a precise updating of $\mathbf{m}(pc)$ and $\mathbf{m}(qc)$. Roughly speaking it would require to reach a steady state in finite time which is impossible with linear ODEs. Thus the second stage consists in trying to make the marking of pc as close as possible to j and the one of qc as close as possible to $l+1-j$.

It consists in two transition pairs depending whether the index i of the current instruction is greater or smaller than j. The first case is illustrated in figure 11 (the other case is similar).

The transition pair $tn_{j,1}, tn_{j,2}$ is activated if both $\mathbf{m}(pn) = j$, $\mathbf{m}(qn) = l+1-j$

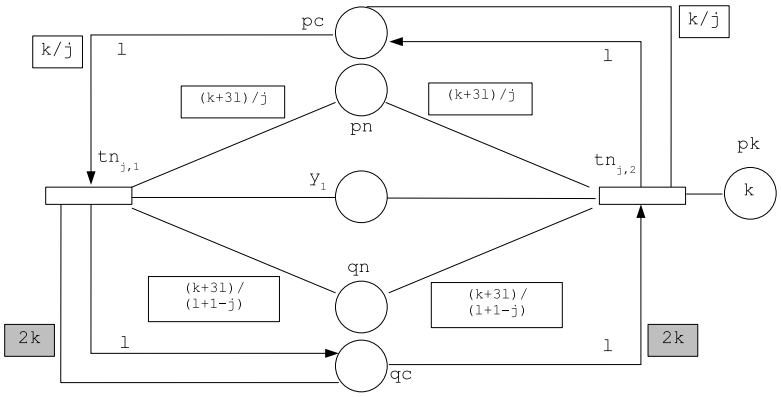

Figure 11: The second stage

and $\mathbf{m}(pc) > j$. If the rate of transition $tn_{j,1}$ was controlled during the whole stage by y_1, pc would loose l tokens. But this means that at the beginning of the stage $\mathbf{m}(pc) > j$ and at the end $\mathbf{m}(pc) \leq j$ which is impossible since $\mathbf{m}(pc)$ must be greater than j in order to trigger the transition pair and thus cannot reach the value j (see our previous remark on linear differential equations). Thus during the second stage *pn must control* the rate of this transition. Since $\mathbf{m}(y_1) \leq k+1$, this means that, at the end of the stage, $(k/j)\mathbf{m}(pc) \leq k+1$ which implies $j \leq m(pc) \leq j+j/k \leq j+l/k$ and consequently $l+1-j-l/k \leq m(qc) \leq l+1-j$ as required by the simulation. The case $i < j$ leads, at the end of the second stage, to $j - l/k \leq m(pc) \leq j$ and consequently $l+1-j \leq m(qc) \leq l+1-j-l/k$.

Theorem 6 is a consequence of our different constructions. The dimension of the associated of ODE is obtained by recalling that the ODE of the clock subnet is 2 and that the following invariants hold in the operating subnet: $\mathbf{m}(pn) + \mathbf{m}(qn) = \mathbf{m}(pc) + \mathbf{m}(qc) = l + 1$. The proof of robustness is omitted.

Theorem 6 *Given a two counter machine $\mathcal{M}$, one can build a TDPN $\mathcal{D}$, with a constant number of places, whose size is linear w.r.t. the machine, whose associated ODE has dimension* 6 *and such that $\mathcal{D}$ robustly simulates $\mathcal{M}$.*

Using a more elaborated simulation of the counters described in [6], one obtains a bounded simulation with an increase of the dimension for the associated ODE. The additional complexity is due to the fact that the simulation of the counters by bounded places is obtained *via* an inverse exponential mapping from values to place markings.

Theorem 7 *Given a two counter machine $\mathcal{M}$, one can build a bounded TDPN $\mathcal{D}$ with a constant number of places, whose size is linear w.r.t. the machine and whose associated ODE has dimension* 14 *such that $\mathcal{D}$ simulates $\mathcal{M}$.*

4.5. Undecidability results

In this section, we apply the simulation results in order to obtain undecidability

results. Proofs are omitted. Note that the status of the marking reachability problem is still open since in the simulation, places pc and qc are not required to take precise values. However the steady-state analysis, a kind of ultimate reachability, is undecidable.

Proposition 4 (Coverability and reachability) *Let $\mathcal{D}$ be a (resp. bounded) TDPN whose associated ODE has dimension at least* 6 *(resp.* 14*), $\mathbf{m}_0, \mathbf{m}_1$ be markings, p be a place and $k \in \mathbb{N}$ then:*

– the problem whether there is a τ such that the trajectory starting at $\mathbf{m}_0$ fulfills $\mathbf{m}(\tau)(p) = k$ is undecidable.

– The problem whether there is a τ such that the trajectory starting at $\mathbf{m}_0$ fulfills $\mathbf{m}(\tau)(p) \geq k$ is undecidable.

– The problem whether there is a τ such that the trajectory starting at $\mathbf{m}_0$ fulfills $\mathbf{m}(\tau) \geq \mathbf{m}_1$ is undecidable.

Proposition 5 (Steady-state analysis) *Let $\mathcal{D}$ be a (resp. bounded) TDPN whose associated ODE has dimension at least* 8 *(resp.* 16*), $\mathbf{m}_0$ be a marking. Then the problem whether the trajectory $\mathbf{m}$ starting at $\mathbf{m}_0$ is such that $\lim_{\tau \to \infty} \mathbf{m}(\tau)$ exists, is undecidable.*

5. Conclusion

In this work, we have introduced continuous Petri nets and we have established efficient algorithms for deadlock-freeness and liveness. Then in a timed framework, we have shown their equivalence with time differentiable Petri nets and we have designed a simulation of counter machines by these models. Using this simulation, we have also proved that marking coverability, submarking reachability and the existence of a steady-state are undecidable. We conjecture that the marking reachability is undecidable and we will try to prove it. In order to obtain decidability results, we also plane to introduce subclasses of continuous Petri nets where the restrictions will be related to both the structure of the net and the associated ODE.

References

1. H. Alla and R. David. "Continuous and hybrid Petri nets," *Journal of Circuits, Systems, and Computers*, 8(1):159–188, 1998.
2. R. Alur and D. Dill. "A theory of timed automata," *Theoretical Computer Science*, 126(2):183–235, 1994.
3. E. Asarin, O. Maler, and A. Pnueli. "Reachability analysis of dynamical systems having piecewise-constant derivatives," *Theoretical Computer Science*, 138-1:35–65, 1995.
4. M.S. Branicky. "Universal computation and other capabilities of hybrid and continuous dynamical systems," *Theoretical Computer Science*, 138(1):67–100, 1995.
5. J.W. Forrester. *Principles of Systems*. Productivity Press, 1968.
6. S. Haddad, L. Recalde, and M. Silva. "On the computational power of Timed Differentiable Petri nets," In E. Asarin and P. Bouyer, editors, *Formal Modeling and Analysis of Timed Systems, 4th Int. Conf. FORMATS 2006*, volume 4202 of *LNCS*, pages 230–244, Paris, France, 2006. Springer.

7. E. Hainry. "Reachability in linear dynamical systems," *Computability in Europe 2008*, Volume 5028 of *LNCS*, pages 241–250, Athens, Greece, 2008. Springer.

8. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. "What's decidable about hybrid automata?," *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

9. J.E. Hopcroft and J.D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, 1969.

10. J. Júlvez, L. Recalde, and M. Silva. "On reachability in autonomous continuous Petri net systems," In W. van der Aalst and E. Best, editors, *Proc. of the* 24^{th} *Int. Conf. on Application and Theory of Petri Nets (ICATPN 2003)*, volume 2679, pages 221–240. Springer, Eindhoven, The Netherlands, June 2003.

11. L. Recalde, S. Haddad and M. Silva, "Continuous Petri Nets: Expressive Power and Decidability Issues" In *Automated Technology for Verification and Analysis, 5th International Symposium, ATVA 2007,*, volume 4762 of *LNCS*, pages 362–377, Tokyo, Japan, 2007. Springer.

12. L. Recalde, E. Teruel, and M. Silva. "Autonomous continuous P/T systems," In S. Donatelli and J. Kleijn, editors, *Application and Theory of Petri Nets 1999*, volume 1639 of *LNCS*, pages 107–126. Springer, 1999.

13. M. Silva. "Introducing Petri nets," In *Practice of Petri Nets in Manufacturing*, pages 1–62. Chapman & Hall, 1993.

14. M. Silva and L. Recalde. "Continuization of timed Petri nets: From performance evaluation to observation and control," In G. Ciardo and P. Darondeau, editors, *Procs. of ICATPN 2005*, volume 3536 of *LNCS*, pages 26–46. Springer, 2005.

15. M. Silva, E. Teruel, and J.M. Colom. "Linear algebraic and linear programming techniques for the analysis of net systems," In G. Rozenberg and W. Reisig, editors, *Lectures in Petri Nets. I: Basic Models*, volume 1491 of *LNCS*, pages 309–373. Springer, 1998.