On the Computational Power of Timed Differentiable Petri Nets

S. Haddad[1], L. Recalde[2], M. Silva[2]

[1] LAMSADE-CNRS UMR 7024, University Paris-Dauphine, France
E-mail: haddad@lamsade.dauphine.fr

[2] GISED, University Zaragossa, Spain
E-mail: {lrecalde | silva}@unizar.es

Abstract. Two kinds of well studied dynamical systems are state and time discrete ones (important in Computer Science) and state and time continuous ones (heavily considered in Automatic Control). Whereas in the discrete setting, the analysis of computational power of models has led to well-known hierarchies, the situation is more confused in the continuous case. A possible way to discriminate between these models is to state whether they can simulate Turing machine (or other equivalent models like counter machines). For instance, it is known that continuous systems described by ordinary differential equations (ODE) have this power. However, since the involved ODE is defined by overlapping local ODEs inside an infinite number of regions, this result has no significant application for differentiable models whose ODE is defined by an explicit representation.

In this work, we considerably strengthen this result by showing that Time Differentiable Petri Nets (TDPN) 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 ODE inside polyhedra. More precisely, we present different simulations in order to fulfill opposite requirements like robustness (allowing some perturbation of the simulation) and boundedness of the simulating net system. Then we establish that the simulation of two counter machines can be performed by a net with a constant number of places, i.e. whose dimension of associated ODE is constant. Afterwards, by modifying the simulation, we prove that marking coverability, submarking reachability and the existence of a steady-state are undecidable. Finally, we study the relations between TDPNs and time continuous Petri nets under infinite server semantics, a formalism with numerous results in control theory. We show that these models are equivalent and we analyse the complexity of the corresponding translations.

Keywords: Dynamic Systems, Time Differentiable Petri Nets, Expressiveness, Simulation, Coverability, Reachability, Steady-state Analysis.

1 Introduction

Hybrid systems. Dynamic systems can be classified depending on the way time is represented. Trajectories of discrete-time systems are obtained by iterating a transition function whereas the ones of continuous-time systems are solutions of a differential equation. When a system includes both continuous and discrete transitions it is called an *hybrid system*. On the one hand, the expressive power of hybrid systems can be strictly greater than the one of Turing machines (see for instance [12]). On the other hand, in restricted models like timed automata [1], several problems including reachability can be checked in a relatively efficient way (i.e. they are $PSPACE$-complete). The frontier between decidability and undecidability in hybrid systems is still an active research topic [8,10,4,11].

Continuous systems. A special kind of hybrid systems where the trajectories are continuous and right-differentiable functions of time have been intensively studied. They are defined by a finite number 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 [2] has piecewise constant derivatives inside regions which are polyhedra and it is Turing equivalent if its space dimension is at least 3 (see also [3,5] for additional expressiveness results).

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 [6], the author shows that differentiable systems in $\mathbb{R}^3$ can simulate Turing machine. This result is a significant contribution to the expressiveness of differentiable systems. However this simulation is somewhat unsatisfactory:

- The 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. So the translation is obtained *from the semantics of the machine* and not from its syntax.
- Thus the effective equations of the ODE are piecewise defined inside *an infinite number regions* which is far beyond the expressiveness of standard ODE formalisms used for the analysis of dynamical systems.

Thus 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. In this work, we (partially) answer this question by showing that Time Differentiable Petri Nets 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 ODEs $\dot{x} = \mathbf{M} \cdot x$ (with $\mathbf{M}$ a matrix) inside polyhedra.

More precisely, we present different simulations in order to fulfill opposite requirements like robustness (allowing some perturbation of the simulation) and boundedness of the simulating net system. Then we establish that the simulation of two counter machines can be performed by a net with a constant number of places, i.e. whose dimension of its associated ODE is constant. Summarizing our results, TDPNs whose associated ODE is in $(\mathbb{R}_{\geq 0})^6$ can robustly simulate Turing machines and bounded TDPNs whose associated ODE is in $[0, K]^{10}$ (for some fixed K) can simulate Turing machines.

Afterwards, by modifying the simulation, we prove that marking coverability, submarking reachability and the existence of a steady-state are undecidable. Finally, we study the relations between TDPNs and time continuous Petri nets under infinite server semantics, a formalism with numerous results in control theory [13]. We show that these models are equivalent and we analyse the complexity of the corresponding translations.

Organisation of the paper. In section 2, we define TDPNs with different examples in order to give intuition about the behaviour of such systems. Then we design a first simulation of counter machines in section 3. We introduce the concept of robust simulation in section 4 and we show that the simulation is robust. Afterwards we design another (non robust) simulation by bounded nets and we transform our two simulations in order to obtain a constant number of places when simulating two counter machines. Finally, we analyse the relation between TDPNs and time continuous Petri nets in section 5. At last, we conclude and give some perspectives to this work.

2 Timed Differentiable Petri Nets

2.1 Definitions

Notations. Let f be a *partial* mapping then $f(x) = \perp$ means that $f(x)$ is undefined. Let $\mathbf{M}$ be a matrix whose domain is $A \times B$, with $A \cap B = \emptyset$ and $a \in A$ (resp. $b \in B$) then $\mathbf{M}(a)$ (resp. $\mathbf{M}(b)$) denotes the vector corresponding to the row a (resp. the column b) of $\mathbf{M}$.

Since *Time Differentiable Petri Nets* (TDPNS) are based on Petri nets, we recall their definition.

Definition 1 (Petri Nets). *A (pure) Petri Net $\mathcal{N} = \langle P, T, \mathbf{C} \rangle$ is defined by:*

- *P, a finite set of places,*
- *T, a finite set of transitions with $P \cap T = \emptyset$,*
- *$\mathbf{C}$, the incidence matrix from $P \times T$ to $\mathbb{N}$, we denote by ${}^\bullet t$ (resp. $t^\bullet$) the set of input places (resp. output places) of t, $\{p \mid \mathbf{C}(p,t) < 0\}$ (resp. $\{p \mid \mathbf{C}(p,t) > 0\}$). $\mathbf{C}(t)$ is called the incidence of t.*

Different semantics may be associated with a Petri net. In a discrete framework, a state $\mathbf{m}$, called a *marking*, is a positive integer vector over the set of places (i.e. an item of $\mathbb{N}^P$) where an unit is called a token. The state change is triggered by transition *firings*. In $\mathbf{m}$, the firing of a transition t with multiplicity $k \in \mathbb{N}_{>0}$ yielding marking $\mathbf{m}' = \mathbf{m} + k\mathbf{C}(t)$ is only possible if $\mathbf{m}'$ is positive. Given m, k_{max} the enabling degree of t, which represents the maximal number of simultaneous firings of t, is defined by: $k_{max} = min(\lfloor \mathbf{m}(p)/(-\mathbf{C}(p,t)) \rfloor \mid p \in {}^\bullet t)$.

In a continuous framework, a marking $\mathbf{m}$, is a positive real vector over the set of places (i.e. an item of $(\mathbb{R}_{\geq 0})^P$). The state change is also triggered by transition firings. However this firing may be any "fraction" say $0 < \alpha$ of the discrete firing yielding $\mathbf{m}' = \mathbf{m} + \alpha\mathbf{C}(t)$, possible if $\mathbf{m}'$ is positive. Here, α_{max} the maximal firing is defined by: $\alpha_{max} = min(\mathbf{m}(p)/(-\mathbf{C}(p,t)) \mid p \in {}^\bullet t)$.

Note that in both models, the choice of the transition firing is non deterministic. In TDPNs, this non determinism is solved by computing at any instant the instantaneous firing rate of every transition and then applying the incidence matrix in order to deduce the infinitesimal variation of the marking. The instantaneous firing rate of transitions depends on the current marking *via* the speed control matrix whose meaning will be detailed in the following definitions.

Definition 2 (Timed Differentiable Petri Nets). *A Timed Differentiable Petri Net $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W} \rangle$ is defined by:*

- *P, a finite set of places,*
- *T, a finite set of transitions with $P \cap T = \emptyset$,*
- *$\mathbf{C}$, the incidence matrix from $P \times T$ to $\mathbb{N}$,*
- *$\mathbf{W}$, the speed control matrix a partial mapping from $P \times T$ to $\mathbb{R}_{>0}$ such that:*
  1. *$\forall t \in T, \exists p \in P, \mathbf{W}(p,t) \neq \perp$*
  2. *$\forall t \in T, \forall p \in P, \mathbf{C}(p,t) < 0 \Rightarrow \mathbf{W}(p,t) \neq \perp$*

The first requirement about $\mathbf{W}$ ensures that the firing rate of any transition may be determined whereas the second one ensures that the marking remains positive since any input place of a transition will control its firing rate. Given a marking $\mathbf{m}$, it remains to determine the instantaneous firing rate of a transition $\mathbf{f}(\mathbf{m})(t)$.

In TDPNs, (when defined) $\mathbf{W}(p,t)$ *weights* the impact of the marking of p on the firing rate of t. Thus we define the instantaneous firing rate similarly to the enabling degree as:

$$\mathbf{f}(\mathbf{m})(t) = min(\mathbf{W}(p,t) \cdot \mathbf{m}(p) \mid \mathbf{W}(p,t) \neq \perp)$$

We are now in position to give semantics to TDPNs.

Definition 3 (Trajectory). *Let $\mathcal{D}$ be a TDPN, then 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:*

$$\dot{\mathbf{m}} = \mathbf{C} \cdot \mathbf{f}(\mathbf{m}) \tag{1}$$

In fact, one can easily prove that if $\mathbf{m}(0)$ is positive, the requirement of positivity is a consequence of the definition of TDPNs. Moreover, one can also prove (by a reduction to the linear equation case) that given an initial marking there is always a single trajectory, i.e. the semantics of a net is well-defined and deterministic.

Equation 1 is particularly simple since it is expressed as a linear equation enlarged with the *min* operator. We introduce the concept of *configurations*: a configuration assigns to a transition, the place that will control its firing rate. Thus the number of configurations is finite. The set of markings corresponding to a configuration is a polyhedron. Inside such a polyhedron, equation 1 becomes a linear differential equation.

Definition 4 (Configuration). *Let $\mathcal{D}$ be a TDPN, then a configuration cf of $\mathcal{D}$ is a mapping from T to P such that $\forall t \in T, \mathbf{W}(cf(t), t) \neq \perp$. Let cf be a configuration, then $[cf]$ denotes the following polyhedron:*

$$[cf] = \{\mathbf{m} \in (\mathbb{R}_{\geq 0})^P \mid \forall t \in T, \forall p \in P,$$

$$\mathbf{W}(p,t) \neq \perp \Rightarrow \mathbf{W}(p,t) \cdot \mathbf{m}(p) \geq \mathbf{W}(cf(t),t) \cdot \mathbf{m}(cf(t))\}$$

By definition, there are $\prod_{t \in T} |\{p \mid \mathbf{W}(p,t) \neq \perp\}| \leq |P|^{|T|}$ configurations. Inside the polyhedron $[cf]$, the differential equation ruling $\mathcal{D}$ becomes linear:

$$\forall p \in P, \dot{\mathbf{m}}(p) = \sum_{t \in T} \mathbf{C}(p,t) \cdot \mathbf{W}(cf(t),t) \cdot \mathbf{m}(cf(t))$$

In the sequel, we use indifferently the word configuration to denote both the mapping cf and the polyhedron $[cf]$.

Graphical notations We extend the graphical notations of Petri net in order to take into account matrix $\mathbf{W}$. A Petri net is a bipartite graph where places are represented by circles (sometimes with their initial marking inside) and transitions by rectangles. An arc denotes a relation between a place and a transition. Note that arcs corresponding to matrix $\mathbf{C}$ are oriented whereas arcs corresponding to matrix $\mathbf{W}$ are unoriented. We describe below the four possible patterns illustrated in figure 1.

- $\mathbf{W}(p,t) = \perp \wedge \mathbf{C}(p,t) > 0$, place p receives tokens from t and does not control its firing rate. There is an oriented arc from t to p labelled by $\mathbf{C}(p,t)$
- $\mathbf{W}(p,t) \neq \perp \wedge \mathbf{C}(p,t) < 0$, place p provides tokens to t. So it must control its firing rate. The unoriented arc between p and t is redundant, so we will not draw it and represent only an oriented arc from p to t both labelled by $-\mathbf{C}(p,t)$ and $\mathbf{W}(p,t)$. In order to distinguish between these two labels, $\mathbf{W}(p,t)$ will always be drawn inside a box.
- $\mathbf{W}(p,t) \neq \perp \wedge \mathbf{C}(p,t) > 0$, place p receives tokens from t and controls its firing rate. There is both an oriented arc from t to p and an unoriented arc between p and t with their corresponding labels.
- $\mathbf{W}(p,t) \neq \perp \wedge \mathbf{C}(p,t) = 0$, place p controls the firing rate of t and t does not modify the marking of p. There is an unoriented arc between p and t.

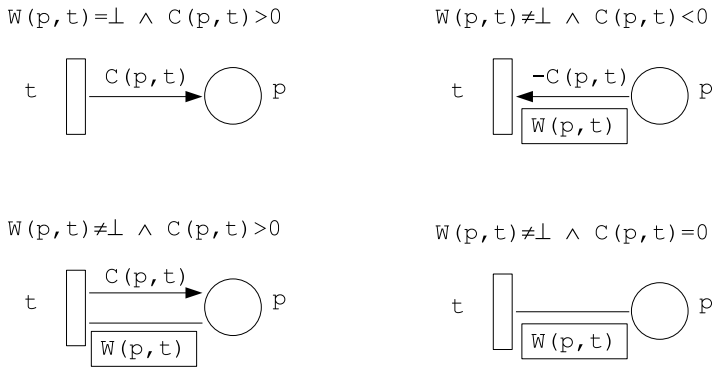

Fig. 1. Graphical notations

We omit labels $\mathbf{C}(p,t)$, $-\mathbf{C}(p,t)$ and $\mathbf{W}(p,t)$ when they are equal to 1.

The first two patterns occur more often in the modelling of realistic systems than the two other ones. In section 5, we will show that the two last patterns may be simulated (with some increasing of the net size).

2.2 Examples

In order to simplify the notations, when writing the differential equations, we use p as a notation for $\mathbf{m}(p)$ (the trajectory projected on p).

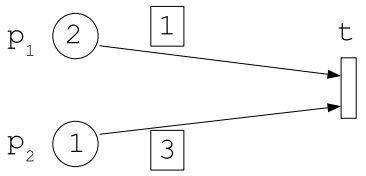

Fig. 2. A marked net switching once

A configuration switch. The net of figure 2 illustrates the main concepts of TDPNs. The firing rate of t is given by $\mathbf{f}(m)(t) = min(\mathbf{m}_1(p), 3\mathbf{m}_2(p))$. Thus equation 1 becomes:

$$\dot{p}_1 = \dot{p}_2 = -min(p_1, 3p_2)$$

There are two configurations $[cf_1] = \{\mathbf{m} \mid \mathbf{m}(p_1) \leq 3\mathbf{m}(p_2)\}$ and $[cf_2] = \{\mathbf{m} \mid \mathbf{m}(p_1 \geq 3\mathbf{m}(p_2)\}$. Inside $[cf_1]$, equation 1 becomes $\dot{p}_1 = \dot{p}_2 = -p_1$ and inside $[cf_2]$, equation 1 becomes $\dot{p}_1 = \dot{p}_2 = -3p_2$.

The behaviour of this net can be straightforwardly analysed. It starts in $[cf_1]$ where it stays until it reaches marking $\mathbf{m} = (3/2, 1/2)$. Then it switches to configuration $[cf_2]$ where it definitely stays and ultimately reaches the steady-state marking $(0, 0)$. This behaviour is illustrated in figure 3.

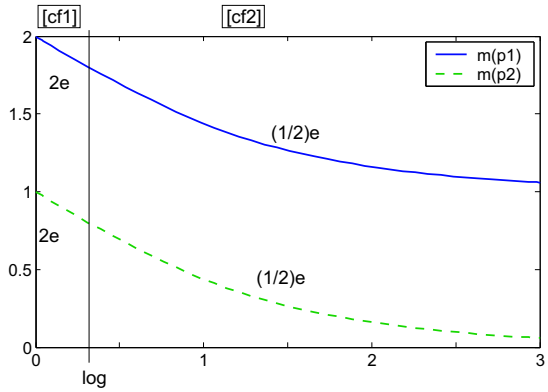

Fig. 3. The behaviour of net of figure 2

Infinitely switching. Let us suppose that a marked TDPN has a steady-state. This steady-state belongs to (at least) one configuration and fulfills the equation for steady-states (i.e. $\mathbf{C} \cdot \mathbf{f}(m) = 0$) which is a linear equation inside a configuration. Thus the identification of possible steady-states is reduced to the identification of such states for every configuration.

One may wonder whether the asymptotical behaviour of a net approaching a steady-state can be reduced to the corresponding behaviour for every configuration. This would be the case if a marked net leading to a steady-state switches between configurations only a finite number of times as does the the net of pevious examples. However as the analysis of the net of figure 4 will show, a net may infinitely switch between configurations while reaching a steady-state.

The ordinary differential equation corresponding to this net is:

$$\begin{aligned}
\dot{x}_1 &= \mathbf{f}(t_4) - \mathbf{f}(t_1) - \mathbf{f}(t) = y_2 - x_1 - \min(x_1, x_2) \\
\dot{x}_2 &= \mathbf{f}(t_2) - \mathbf{f}(t_3) - \mathbf{f}(t) = y_1 - x_2 - \min(x_1, x_2) \\
\dot{y}_1 &= \mathbf{f}(t_1) - \mathbf{f}(t_2) = x_1 - y_1 \\
\dot{y}_2 &= \mathbf{f}(t_3) - \mathbf{f}(t_4) = x_2 - y_2
\end{aligned}$$

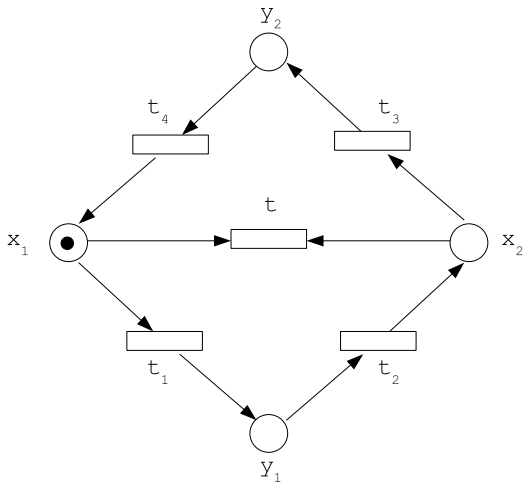

Fig. 4. A net infinitely switching

The two configurations of this net are $[cf_1] = \{\mathbf{m} \mid \mathbf{m}(x_1) \geq \mathbf{m}(x_2)\}$ and $[cf_2] = \{\mathbf{m} \mid \mathbf{m}(x_1) \leq \mathbf{m}(x_2)\}$. Its evolution can be observed in Fig. 5.

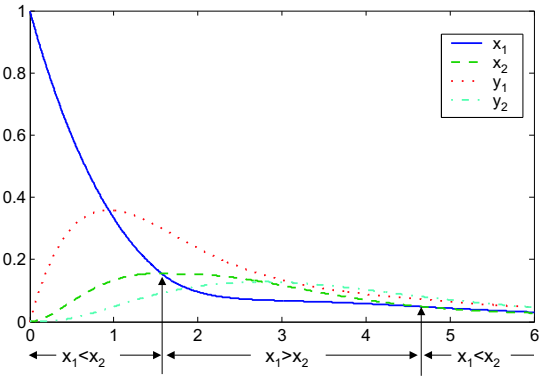

Fig. 5. Evolution of the system described by the net of fig. 4

Observation. *The net of figure 4 reaches a steady state while infinitely switching between configurations.*

Proof. Let us write some behavioural equations related to this net. We note $\delta_x = \mathbf{m}(x_1) - \mathbf{m}(x_2)$ and $\delta_y = \mathbf{m}(y_1) - \mathbf{m}(y_2)$. Then:

$$\dot{\delta}_x = -\delta_x - \delta_y$$

$$\dot{\delta}_y = \delta_x - \delta_y$$

Taking into account the initial conditions, these equations yield to:

$$\delta_x = e^{-\tau} cos(\tau)$$

$$\delta_y = e^{-\tau} sin(\tau)$$

Thus the net infinitely switches between configuration $\mathbf{m}(x_1) \geq \mathbf{m}(x_2)$ and $\mathbf{m}(x_1) \leq \mathbf{m}(x_2)$.

It remains to show it (asymptotically) reaches a steady-state.
Let us note $s = \mathbf{m}(x_1) + \mathbf{m}(x_2) + \mathbf{m}(y_1) + \mathbf{m}(y_2)$.
One has $\dot{s} = -2\inf(x_1, x_2)$. Thus s is decreasing, let us note l its limit.
We show that $l = 0$.
Assume that $l > 0$.
Since $\lim_{\tau\to\infty} \delta_x(\tau) = 0$, there is a n_0 such that:
$\forall n \geq n_0, |\mathbf{m}(x_1)((2n+1)\pi) - \mathbf{m}(x_2)((2n+1)\pi)| \leq l/4$.
Since $\mathbf{m}(y_1)((2n+1)\pi) = \mathbf{m}(y_2)((2n+1)\pi)$,
we conclude that:
$\forall n \geq n_0$, either $\mathbf{m}(y_1)((2n+1)\pi) = \mathbf{m}(y_2)((2n+1)\pi) \geq l/4$
or $min(\mathbf{m}(x_1)((2n+1)\pi), \mathbf{m}(x_2)((2n+1)\pi)) \geq l/8$.
We note $\tau_0 = (2n+1)\pi$ and τ any value greater or equal than τ_0.

Case $\mathbf{m}(y_1)(\tau_0) = \mathbf{m}(y_2)(\tau_0) \geq l/4$
The following inequation holds: $\dot{\mathbf{m}}(y_i) \geq -\mathbf{m}(y_i)$.
Thus $\mathbf{m}(y_i)(\tau) \geq (l/4)e^{-(\tau-\tau_0)}$ and $\dot{\mathbf{m}}(x_i) \geq (l/4)e^{-(\tau-\tau_0)} - 2\mathbf{m}(x_i)$.
Consequently, $\mathbf{m}(x_i)(\tau) \geq (\mathbf{m}(x_i)(\tau_0) + (l/4)(e^{\tau-\tau_0} - 1))e^{-2(\tau-\tau_0)}$
$\geq (l/4)(e^{\tau-\tau_0} - 1)e^{-2(\tau-\tau_0)}$.
Finally, $s(\tau_0) - s(\tau_0 + 2\pi) \geq \int_0^{2\pi} (l/2)(e^v - 1)e^{-2v} dv$.
Call this last value d_1.

Case $min(\mathbf{m}(x_1)(\tau_0), \mathbf{m}(x_2)(\tau_0)) \geq l/8$
The following inequation holds: $\dot{\mathbf{m}}(x_i) \geq -2\mathbf{m}(x_i)$.
Consequently, $\mathbf{m}(x_i)(\tau) \geq (l/8)e^{-2(\tau-\tau_0)}$
Finally, $s(\tau_0) - s(\tau_0 + 2\pi) \geq \int_0^{2\pi} (l/4)e^{-2v} dv$.
Call this last value d_2.
Summarizing, one obtains $s((2n+1)\pi) \leq s((2n_0+1)\pi) - (n - n_0)min(d_1, d_2)$
in contradiction with positivity of s.

Periodic behaviour. The last example shows that the asymptotic behaviour of a net may strongly depend on the configurations it reaches.

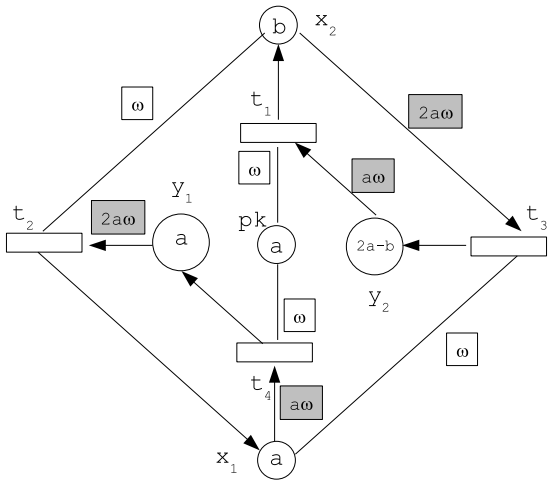

Fig. 6. A periodic TDPN

The ordinary differential equation corresponding to the net of Figure 6 is (note that place *pk* holds a constant number of tokens):

$$\begin{aligned}
\dot{x}_1 &= \mathbf{f}(t_2) - \mathbf{f}(t_4) = \min\{\omega \cdot x_2, 2a\omega \cdot y_1\} - \min\{a\omega \cdot x_1, \omega\} \\
\dot{x}_2 &= \mathbf{f}(t_1) - \mathbf{f}(t_3) = \min\{a\omega \cdot y_2, \omega\} - \min\{2a\omega \cdot x_2, \omega \cdot x_1\} \\
\dot{y}_1 &= \mathbf{f}(t_4) - \mathbf{f}(t_2) = \min\{a\omega \cdot x_1, \omega\} - \min\{\omega \cdot x_2, 2a\omega \cdot y_1\} \\
\dot{y}_2 &= \mathbf{f}(t_3) - \mathbf{f}(t_1) = \min\{2a\omega \cdot x_2, \omega \cdot x_1\} - \min\{a\omega \cdot y_2, \omega\}
\end{aligned}$$

However, it can be observed that $y_1 + x_1$ and $y_2 + x_2$ are constant. Hence the system (with the initial condition described by the marking in the figure) is equivalent to:

$$\begin{aligned}
\dot{x}_1 &= \min\{\omega \cdot x_2, 2a\omega \cdot y_1\} - \min\{a\omega \cdot x_1, \omega \cdot a\} \\
\dot{x}_2 &= \min\{a\omega \cdot y_2, \omega \cdot a\} - \min\{2a\omega \cdot x_2, \omega \cdot x_1\} \\
y_1 &= 2a - x_1 \\
y_2 &= 2a - x_2
\end{aligned}$$

This corresponds in fact to a set of sixteen configurations. Let us solve the differential system with $1 \leq a \leq b \leq 2a - 1$. The linear system that applies

initially is:

$$\begin{aligned}
\dot{x}_1 &= \omega \cdot x_2 - \omega \cdot a \\
\dot{x}_2 &= \omega \cdot a - \omega \cdot x_1 \\
y_1 &= 2a - x_1 \\
y_2 &= 2a - x_2
\end{aligned}$$

In figure 6, we have represented the "inactive" items of matrix $\mathbf{W}$ in a shadowed box. In the sequel, we use this convention when it will be relevant. The solution of this system is:

$$\begin{aligned}
x_1(\tau) &= a + (b-a)\sin(\omega \cdot \tau) \\
x_2(\tau) &= a + (b-a)\cos(\omega \cdot \tau) \\
y_1(\tau) &= a - (b-a)\sin(\omega \cdot \tau) \\
y_2(\tau) &= a - (b-a)\cos(\omega \cdot \tau)
\end{aligned}$$

This trajectory stays infinitely in the initial configuration and consequently it is the behaviour of the net. However for other initial conditions like $a = 1, b = 2$, configuration switches happen before a steady state is ultimately reached, as can be observed in figure 7.

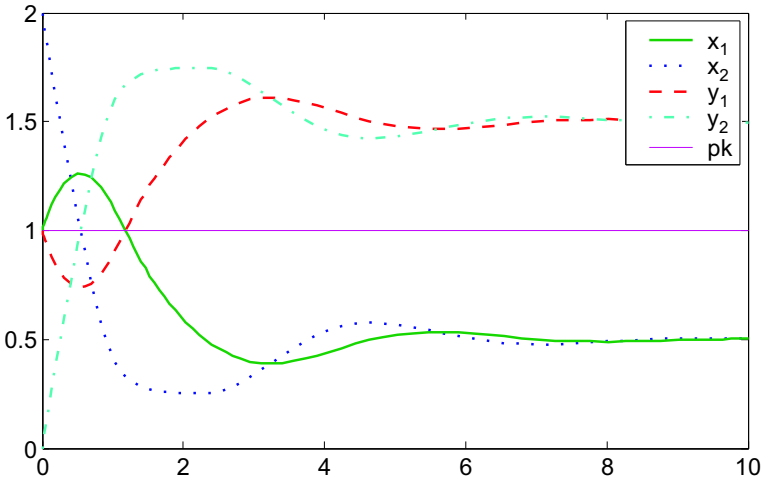

Fig. 7. Evolution of the system described by the net of Fig. 6 with $a = 1, b = 2$

Remark. This example illustrates the fact that the dimension of the ODE may be strictly smaller than the number of places. Indeed, the existence of a linear

invariant such $\sum_{p \in P} \mathbf{m}(p) = cst$ decreases by one unit the number of dimensions. Otherwise stated, the dimension of the ODE is not $|P|$ but $\mathbf{rank}(\mathbf{C})$. In the example, $|P| = 5$ and $\mathbf{rank}(\mathbf{C}) = 2$.

3 A first simulation

3.1 Preliminaries

Counter machines. We will simulate counter machines which are equivalent to Turing machines. A counter machine has positive integer counters, (note that two counters are enough for the equivalence [9]). Its 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):

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

In order to simplify the simulation, we split the test instruction as follows:

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

Furthermore, w.l.o.g. we assume that the (possible) successor(s) of an instruction is (are) always different from it.

Dynamical systems and simulation. The (deterministic) dynamical systems $(X, \mathcal{T}, f)$ we consider are defined by a state space X, a time space $\mathcal{T}$ ($\mathcal{T}$ is either $\mathbb{N}$ or $\mathbb{R}_{\geq 0}$) and 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 a discrete system $X \subseteq \mathbb{N}^d$ for some d and $\mathcal{T} = \mathbb{N}$ whereas in a continous system $X \subseteq (\mathbb{R}_{\geq 0})^d$ for some d and $\mathcal{T} = \mathbb{R}_{\geq 0}$.

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. The requirements of the next definition ensure 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 that would have been reached after n steps.

Definition 5. *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:*

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

3.2 Basic principles of the simulation

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

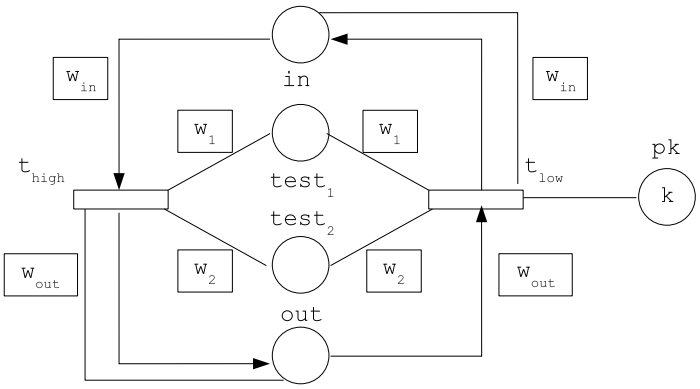

Fig. 8. A transition pair

Let us examine transitions t_{high} and t_{low} of figure 8. 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 will always be k. So we conclude that:

- 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 6 that we will call in the sequel the *clock* subnet and another subnet depending on the counter machine. We call this latter subnet, 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 9 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.

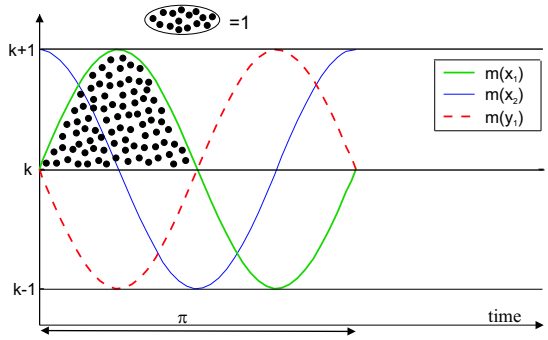

Fig. 9. The behaviour of the clock subnet

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.

At last, note that place pk occurring in the transition pairs is the constant place of the clock subnet.

Specialisation of the transition pairs pattern. Using places x_1 and y_1, we specialise transition pairs as illustrated in figure 10. 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 $\mathbf{m}(x_1)$, we know that no tokens transfer will occur during the second part 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 9) exactly n tokens will be transfered from in to out.
- In the other cases, 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. Thus in some marking $\mathbf{m}$, we say that a place is *active* if $\mathbf{m}(p) \geq k + 1$ and *inactive* if $\mathbf{m}(p) \leq k$. 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}.

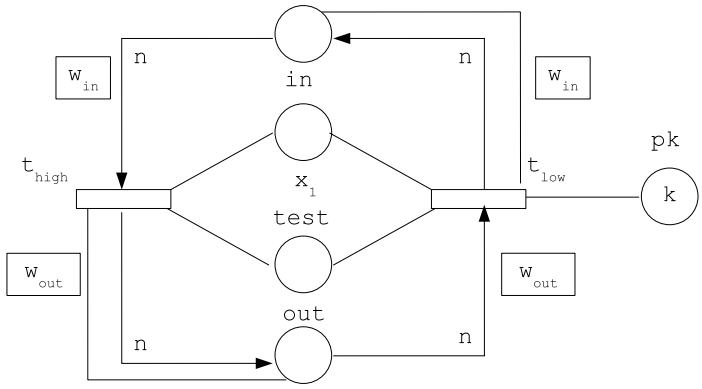

Fig. 10. A specialised transition pair

3.3 The operating subnet

Places of the operating subnet and the simulation mapping. Let us suppose that the program with counters has l instructions $\{\mathtt{I}_1, \ldots, \mathtt{I}_l\}$ and two counters $\{\mathtt{cpt}_1, \mathtt{cpt}_2\}$. The operating subnet will have the following places: $\bigcup_{1 \leq i \leq l} \{pc_i, pn_i\} \cup \{c_1, c_2\}$.

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 (note that this is independent from the state of the machine).
- $\mathbf{m}(pn_i) = k + 2$ and $\mathbf{m}(pc_i) \in [k + 2, k + 11/4]$, note that these places are active.
- $\forall 1 \leq i' \neq i \leq l, \mathbf{m}(pn_{i'}) = k-1$ and $\forall 1 \leq i' \neq i \leq l, \mathbf{m}(pc_{i'}) \in [k-1, k-1/4]$, note that these places are inactive.
- $\mathbf{m}(c_1) = k - 1 + 3v_1, \mathbf{m}(c_2) = k - 1 + 3v_2$, thus we choose a affine transformation of the counter values. Note that if such a value is 0 then the corresponding place is inactive whereas if it is non null, the place is active.

Thus the difference between the marking of an active pn_i and an inactive pn_j is 3 as is the difference between the marking interval for an active pc_i and an inactive pc_j. We will progressively explain the choices relative to this mapping, during the presentation of the operating subnet transitions.

Finally note that k will be chosen enough large to fulfill additional constraints when needed.

Principle of the instruction simulation. The simulation of an instruction $\mathtt{I_i}$ will take 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 $\mathbf{m}(pc_i) \geq k+1$ and performs the following tasks:

- decrementing pn_i (by 3) while incrementing pn_j (by 3) where $\mathtt{I_j}$ is the next instruction. If $\mathtt{I_i}$ is a conditional jump, this involves to find the appropriate j. The marking of pn_i will vary from $k+2$ to $k-1$ and *vice versa*,
- updating the counters depending on the instruction.

The second stage is triggered by $\mathbf{m}(pn_j) \geq k+1$ and performs the following tasks:

- incrementing pc_j (by 3) and decrementing pc_i by a variable value in such a way that their marking still belong to the intervals associated with the simulation mapping,
- by a side effect, possibly decrementing some inactive $pc_{i'}$ but not below $k-1$.

First stage: simulation of an unconditional jump. This part of the simulation applies to both an unconditional jump, an incrementation and a decrementation. We explain in the next paragraph how the counter updates are performed. For this kind of instructions, the next instruction say $\mathtt{I_j}$ is *a priori* known.

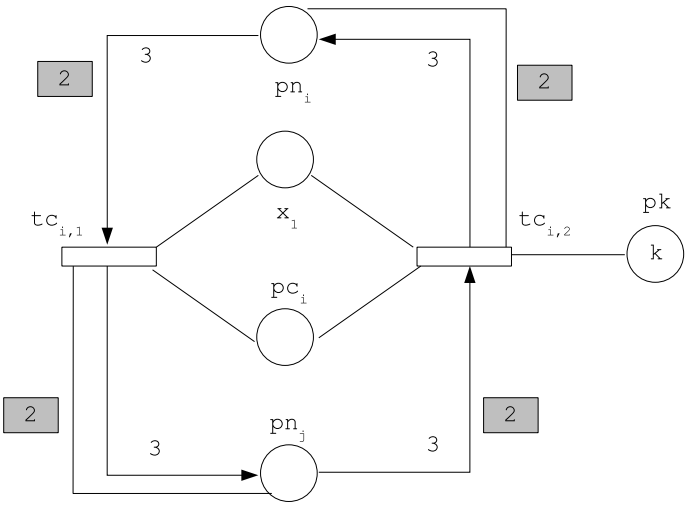

Fig. 11. The first stage of an unconditional jump

This subnet consists in a transition pair which transfers, during the first part of the cycle (due to the place x_1), 3 tokens from pn_i to pn_j iff pc_i is active (i.e. if the simulated instruction is $\texttt{I}_\texttt{i}$).

Note that the value $\mathbf{W}(pn_i, tc_{i,1}) = \mathbf{W}(pn_i, tc_{i,2}) = \mathbf{W}(pn_j, tc_{i,1})$ $= \mathbf{W}(pn_j, tc_{i,2}) = 2$ ensures that places pn_i and pn_j will not control these transitions (since $2(k-1) \geq k+2$ for k enough large).

First stage: counter updates. The update of counters does not raise any particular problem as illustrated in the figure 12. We connect place c_u to the previous transition pair depending whether it is an incrementation or a decrementation. During the first part of the cycle, the place c_u receives or looses the same quantity of tokens as the one received by pn_j, i.e. 3. Again $\mathbf{W}(c_u, tc_{i,1}) = \mathbf{W}(c_u, tc_{i,2}) = 2$ ensures that c_u will not determine the firing rate of the transitions.

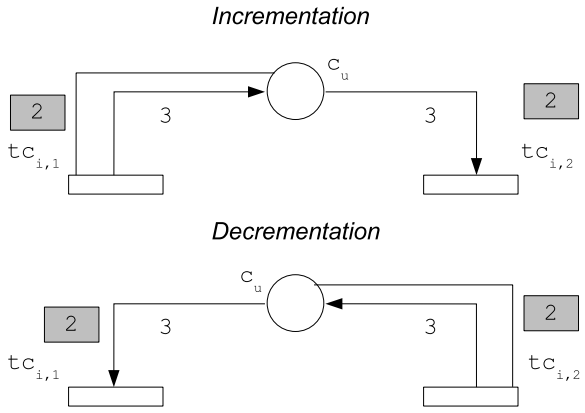

Fig. 12. Counter updates

First stage: simulation of a conditional jump. The first stage for simulating the instruction $\texttt{I}_\texttt{i}$: `if` $\texttt{cpt}_\texttt{u} = 0$ `then goto` $\texttt{I}_\texttt{j}$ `else goto` $\texttt{I}_{\texttt{j}'}$; is illustrated in figure 13.

It consists in two transition pairs. Pair $tc_{i,1}, tc_{i,2}$ mimics the first stage of an unconditional jump from $\texttt{I}_\texttt{i}$ to $\texttt{I}_\texttt{j}$. It will transfer during the first part of the cycle 3 tokens from pn_i to pn_j. Pair $tc_{i,3}, tc_{i,4}$ is triggered if both pc_i and c_u are active (i.e. the counter $\texttt{cpt}_\texttt{u}$ is non null). If it is the case it will transfer 3 tokens from pn_j to $pn_{j'}$. Thus:

– If $\mathbf{m}(c_u) = k-1$ then only the first pair is triggered and 3 tokens will be transfered from pn_i to pn_j.

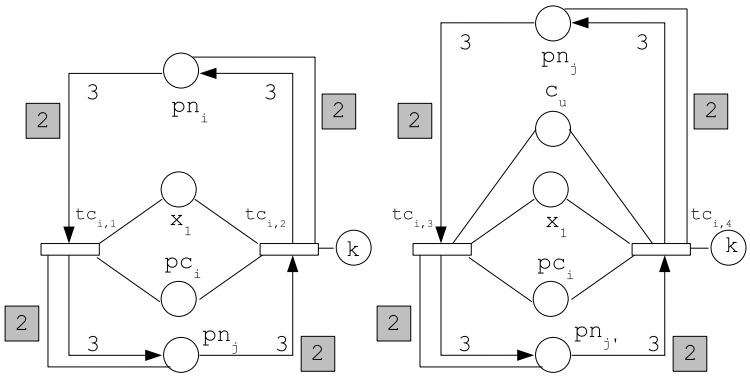

Fig. 13. The first stage of a conditional jump

– otherwise $\mathbf{m}(c_u) \geq k + 2$, the two pairs are simultaneously triggered and 3 tokens will be transfered from pn_i to pn_j and from pn_j to $pn_{j'}$. So the marking of pn_i looses 3 tokens, pn_j is unchanged and $pn_{j'}$ receives 3 tokens as desired.

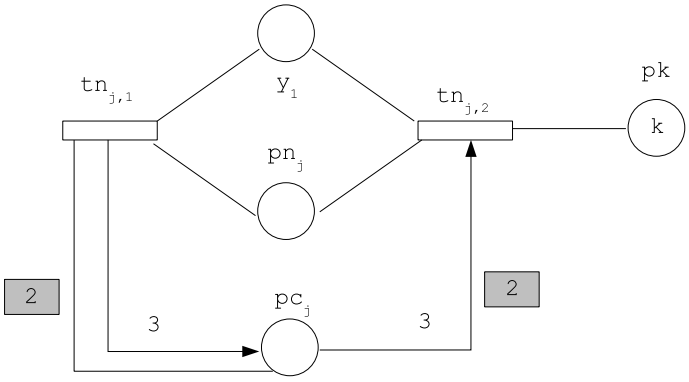

Fig. 14. The second stage: incrementing pc_j

The second stage. The second stage consists in incrementing pc_j and decrementing pc_i. The first operation does not raise any particular difficulty and is illustrated figure 14.

Pair $tn_{j,1}, tn_{j,2}$ is triggered by place pn_j which has become active during the first stage (note that the stage is controlled by y_1). In fact, this pair is only related to the "next" instruction $\mathtt{I_j}$. It will produce exactly 3 tokens to pc_j.

The decrementation of pc_i is the difficult part of the simulation. Indeed, different instructions may jump to $\mathtt{I_j}$. So the transition pair of figure 15 is triggered both by pn_j and pc_i. Note that we create such transitions for every pair $\mathtt{I_i}, \mathtt{I_j}$ of instructions such that $\mathtt{I_j}$ possibly follows $\mathtt{I_i}$. Let us explain the weight $\mathbf{W}(pc_i) = k/(k-1)$: with such a weight the pair is only triggered if $\mathbf{m}(pc_i) \geq k-1$ and then if $\mathtt{I_j}$ is reached by another instruction than $\mathtt{I_i}$, $m(pc_i)$ could decrease but not below $k-1$ as it is required for the simulation.

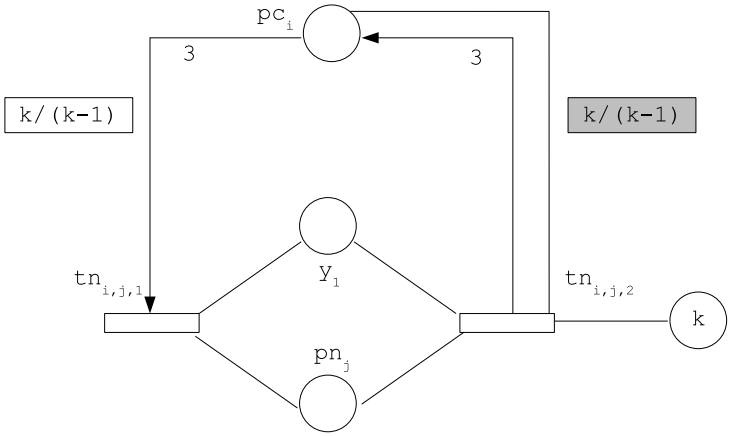

Fig. 15. The second stage: decrementing pc_i

However this additional control has a drawback since if $\mathbf{m}(pc_i) = k+2$ at the beginning of the second part of the cycle, this place will control the firing of $tn_{i,j,1}$ during this part as can been observed in figure 16. Thus pc_i will loose less than 3 tokens. In this figure, we have drawn three curves related to $(k/(k-1))\mathbf{m}(pc_i)$:

- the first one which finishes as a dotted line would be $(k/(k-1))\mathbf{m}(pc_i)$ with $m(pc_i) = k+2$ at the beginning of the stage, *if* $\mathbf{m}(y_1)$ *had controlled the firing rate during the whole stage*, and then 3 tokens would have been transfered;
- the second one which is identical to the first one at the beginning is the real $(k/(k-1))\mathbf{m}(pc_i)$ with $m(pc_i) = k+2$ at the beginning of the stage, and less 3 tokens have been transfered;
- the third one is $(k/(k-1))\mathbf{m}(pc_i)$ with $m(pc_i) = k+11/4$ at the beginning of the stage. We claim that this curve does not meet $\mathbf{m}(y_1)$ and then exactly 3 tokens have been transfered.

Proof of the claim. In order to prove it we write the equation of $\mathbf{m}(pc_i)$ when the firing rate is always controlled by $\mathbf{m}(y_1)$ and we verify that $(k/(k-1))\mathbf{m}(pc_i) \geq \mathbf{m}(y_1)$. during the whole stage.

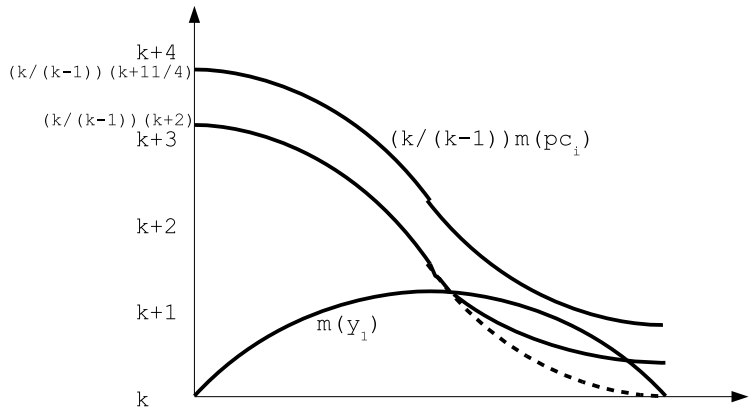

Fig. 16. Which place does control $tn_{i,j,1}$?

We choose the beginning of the second part as the origin of time and study functions during the interval $[0, \pi/2]$.

- $(k/(k-1))\mathbf{m}(pc_i)(\tau) = (k/(k-1))(k + 5/4 + (3/2)cos(2\tau)$
  $\geq (1 + 1/k)(k + 5/4 + (3/2)cos(2\tau)$
  $= (k + 9/4 + (3/2)cos(2\tau) + (1/k)(5/4 + (3/2)cos(2\tau))$
  $\geq k + (3 + \sqrt{2})/2 + (3/2)cos(2\tau)$ for k enough large.
  Let us note $\varphi(\tau) = k + (3 + \sqrt{2})/2 + (3/2)cos(2\tau)$,
  then $\varphi(3\pi/8) = k + (3 + \sqrt{2})/2 - 3\sqrt{2}/4 \geq k + 1$
  and $\varphi(\pi/2) = k + (3 + \sqrt{2})/2 - 3/2 = k + \sqrt{2}/2$.
  Since $\varphi(\tau)$ is decreasing, we conclude that:
  $\forall 0 \leq \tau \leq 3\pi/8, \varphi(\tau) \geq k + 1$
  and $\forall 3\pi/8 \leq \tau \leq \pi/2, \varphi(\tau) \geq k + \sqrt{2}/2$

- $\mathbf{m}(y_1)(\tau) = k + sin(2\tau)$, thus $\forall 0 \leq \tau \leq \pi/2, \mathbf{m}(y_1)(\tau) \leq k + 1$.
  Furthermore, $\mathbf{m}(y_1)(3\pi/8) = k + \sqrt{2}/2$.
  Since $\mathbf{m}(y_1)$ is decreasing in $[3\pi/8, \pi/2]$, we conclude that:
  $\forall 3\pi/8 \leq \tau \leq \pi/2, \mathbf{m}(y_1)(\tau) \leq k + \sqrt{2}/2$

So the claim is established.

We illustrate in figure 17 the evolution of $\mathbf{m}(pc_i)$ at observation epochs. Note that the dotted lines do not represent the evolution of marking but just bind the corresponding values between observation epochs.

The following theorem is a direct consequence of the correctness of our construction.

Theorem 1. *Given a counter machine $\mathcal{M}$, one can build a TCPN $\mathcal{D}$ whose size is linear w.r.t. the machine such that $\mathcal{D}$ simulates $\mathcal{M}$.*

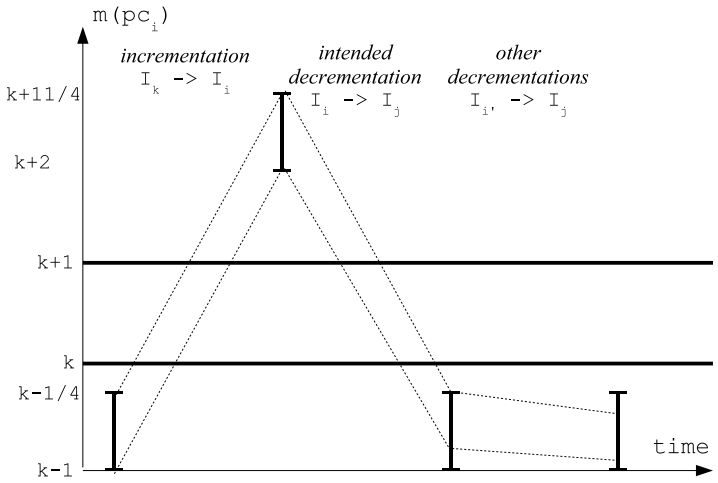

Fig. 17. Evolution of $\mathbf{m}(pc_{i,j,1})$ at observation epochs

4 Refining the simulation

4.1 Robust simulation

In order to introduce robust simulation, we refine the notion of simulation. First, we consider a *two-level* dynamical system $(Y = Y_1 \times Y_2, \mathbb{R}_{\geq 0}, g)$ 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 6. *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:*

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

The critical remark here is 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. The robustness of the simulation is related to pertubations of the function ϕ and to the time of observations. In the following definition, *dist* is the distance related to norm $||_\infty$ (but the definition is essentially independent from the choice of the norm).

Definition 7. *A simulation (by a two-level system) is* robust *iff there exists* $\delta, \epsilon \in \mathbb{R}_{>0}$ *such that:*

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

Thus, if the simulation is robust, starting with an initial 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.

Theorem 2. *Given a counter machine $\mathcal{M}$, one can build a TDPN $\mathcal{D}$ whose size is linear w.r.t. the machine such that $\mathcal{D}$ robustly simulates $\mathcal{M}$.*

Proof. We prove that our simulation is robust. Note that this simulation is a two-level simulation with places of kind pc_i, pn_i, c_u corresponding to the simulation mapping. We choose $\delta = min(1/4, 1/24(k+1)|T|)$ and $\epsilon = 1/2$.

First note that due to our modelling, two markings simulating different configurations x and x' of the machine have at least one place where they differ by at least $3/2(> 1)$ tokens. So ϵ fulfills the first requirement of robustness.

The behaviour of the net is driven by firing rate differences between transition pairs. So this behaviour depends on the "activity" of places pc_i, pn_i, c_u in the following way: an inactive place may have any marking in $[k - 5/4, k]$ and an active behaviour may have a marking in in $[k+7/4, k+3]$. Indeed, the important point is that an inactive place has a marking $m \leq k$ and an active place has a marking $m \geq k + 1$.

Thus a initial perturbation in $[0, 1/4]$, will yield a perturbation at observation epochs which may vary but will be always in $[0, 1/4]$.

Since the behaviour of the net is ruled by a differential equation, let us bound $|\dot{\mathbf{m}}|$. First note that $\forall p, t, |C(p, t)| \leq 3$. Then a simple examination of the transitions of the net shows that $\forall t, \mathbf{f}(\mathbf{m})(t) \leq 2(k+1)$. Thus $|\dot{\mathbf{m}}| \leq 6(k+1)|T|$.

Thus a perturbation of the observation epoch with a value $\leq 1/24(k+1)|T|$ leads to a pertubation of the marking with a value $\leq 1/4$.

Summing the two perturbations yields the result.

4.2 Simulation with a bounded TDPN

In this paragraph, we modify our simulation in order to obtain a bounded net. Note that bounded and robust simulations of an infinite-state system are incompatible for obvious reasons. Our current net is unbounded due to the way we model the counters. So we change their management.

First we will build a new lazy machine $\mathcal{M}'$ from the original one $\mathcal{M}$. We multiply by 4 the number of intructions, i.e. we create three instructions $\mathtt{A_i}$: `goto` $\mathtt{B_i}$;, $\mathtt{B_i}$: `goto` $\mathtt{C_i}$; and $\mathtt{A_i}$: `goto` $\mathtt{I_i}$; per instruction $\mathtt{I_i}$. Then we modify every label in the original instructions by substituting $\mathtt{A_i}$ to $\mathtt{I_i}$. $\mathcal{M}$ and $\mathcal{M}'$ are equivalent from a simulation point of view since they perform the same computation except that four instructions of $\mathcal{M}'$ do what does a single of instruction of $\mathcal{M}$.

We then duplicate the operating subnet ($\mathcal{D}'^{(1)}$ and $\mathcal{D}'^{(2)}$) to simulate $\mathcal{M}$ *via* $\mathcal{M}'$. The only difference between the subnets is that $\mathcal{D}'^{(1)}$ simulates $\mathtt{I_i}$ of $\mathcal{M}$ by simulating $\mathtt{I_i}$ of $\mathcal{M}'$ whereas $\mathcal{N}'^{(2)}$ simulates $\mathtt{I_i}$ of $\mathcal{M}$ by simulating $\mathtt{B_i}$ of $\mathcal{M}'$.

This yields the scheduling given in figure 18. We will also use the superscripts $^{(1)}$ and $^{(2)}$ to distinguish between places of the two subnets.

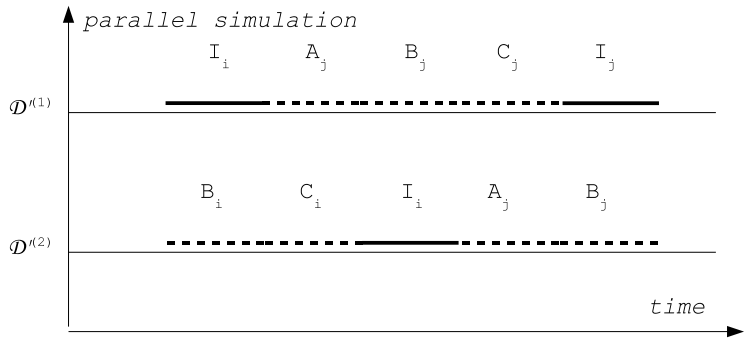

Fig. 18. Interleaving two simulations

In each subnet, we create a place $d_u^{(s)}$ $(s = 1, 2)$ in addition to $c_u^{(s)}$, to model the counter $\mathtt{cpt_u}$. Then we change our counter updates in such a way that when a counter $\mathtt{cpt_u}$ is equal to v then $\mathbf{m}(c_u^{(s)}) = (k+2) - 2(1/2)^v$ and $\mathbf{m}(d_u^{(s)}) = (k-1) + 2(1/2)^v$.

Hence if $\mathtt{cpt_u} = \mathtt{0}$ then $\mathbf{m}(c_u^{(s)}) = k$ and if $\mathtt{cpt_u} \geq \mathtt{1}$ then $\mathbf{m}(c_u^{(s)}) \geq k+1$ as required for the correctness of the simulation of the conditional jump.

It remains to describe the handling of incrementations and decrementations. Note that the main difficulty is that the decrement (or increment) depends on the current value of the simulated counter. If $\mathtt{cpt_u} = \mathtt{v}$ and we increment the counter, then we must produce (resp. consume) $(1/2)^v$ tokens in $c_u^{(s)}$ (resp. in $d_u^{(s)}$). If $\mathtt{cpt_u} = \mathtt{v}$ and we decrement the counter, then we must consume (resp. produce) $(1/2)^{v+1}$ tokens in $c_u^{(s)}$ (resp. in $d_u^{(s)}$).

Let us observe the evolution of marking pn_i in the first simulation (see figure 19) when one simulates the execution of instruction $\mathtt{I_i}$. In the first part of a cycle it raises from $k-1$ to $k+2$, then holds this value during the second part and decreases in the first part of the next cycle to k. The increasing and the decreasing are not linear but they are *symmetrical*. Thus the mottled area of figure 19 is proportional to the difference between c_u and $k+2$ (equal to the difference between d_u and $k-1$). We emphasize the fact that neither the upper part of this area nor its lower part are proportional to the difference.

The subnets of figure 20 and figure 21 exploit this feature to simulate an incrementation of the counter. $\mathtt{I_i}$ is an incrementation of the counter $\mathtt{cpt_u}$. Let us detail the behaviour of the former subnet.

This subnet has two transition pairs $inc_{i,1}, inc_{i,2}$ and $inc_{i,3}, inc_{i,4}$. The firing rate of $inc_{i,1}$ is $(1/(4\pi))min(\mathbf{m}(c_u^{(2)}), \mathbf{m}(pn_i^{(1)}))$ (note again that due to their

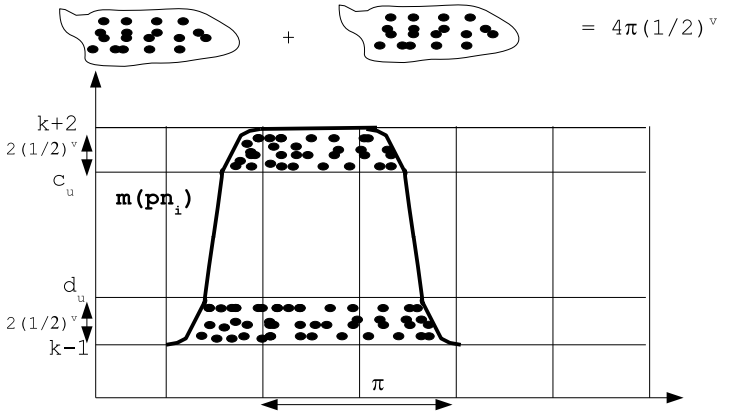

Fig. 19. A way to obtain a "proportional" increment

speed control equal to 1, places $c_u^{(1)}$ and $c_u^{(1)}$ do not determine this rate). The rate of $inc_{i,2}$ is $(1/(4\pi))\mathbf{m}(pn_i^{(1)})$. Thus they have different speed as long as $\mathbf{m}(pn_i^{(1)}) > \mathbf{m}(c_u^{(2)})$. So their effect corresponds to the upper part of the mottled area of figure 19.

The rate of $inc_{i,3}$ is $(1/(4\pi))min(\mathbf{m}(d_u^{(2)}), \mathbf{m}(pn_i^{(1)}))$. The rate of $inc_{i,4}$ is $(k-1)/(4\pi)$. So their effect corresponds to the lower part of the mottled area of figure 19.

The scaling factor $1/4\pi$ ensures that $1/2(\mathbf{m}(d_u^{(2)}) - (k-1))$ have been transfered from $\mathbf{m}(d_u^{(1)})$ to $\mathbf{m}(c_u^{(1)})$.

The net of figure 21 behaves similarly except that since $\mathbf{m}(c_u^{(1)})$ and $\mathbf{m}(d_u^{(1)})$ have their new value the scaling factor must be doubled in order to transfer the same amount of tokens from $\mathbf{m}(d_u^{(2)})$ to $\mathbf{m}(c_u^{(2)})$.

The decrementation simulation follows the same pattern except that the scaling factor of the first stage is doubled w.r.t. the first stage of the incrementation whereas the scaling factor of the second stage is divided by two w.r.t. the second stage of the incrementation.

It remains to show that the places of $\mathcal{D}'^{(2)}$ modelling the counter $\mathtt{cpt_u}$ are not modified during the simulation of the $\mathtt{I_i}$ in $\mathcal{D}'^{(1)}$ and *vice versa*. Looking at figure 22 we see that this is the case as the instruction simulations are translated and surrounded by "no-op" instructions which do not modify the counters.

The correctness of this simulation yields the following theorem.

Theorem 3. *Given a counter machine $\mathcal{M}$, one can build a bounded TDPN $\mathcal{D}$ whose size is linear w.r.t. the machine such that $\mathcal{D}$ simulates $\mathcal{M}$.*

Note that, the speed control matrix of this net has non rational items contrary to the net of the first simulation. They belong to the field $\mathbb{Q}[\pi]$. It is an

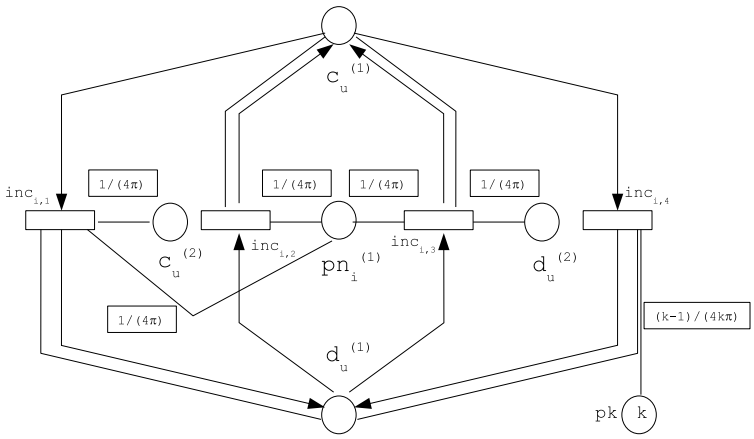

Fig. 20. Incrementing a counter (first stage)

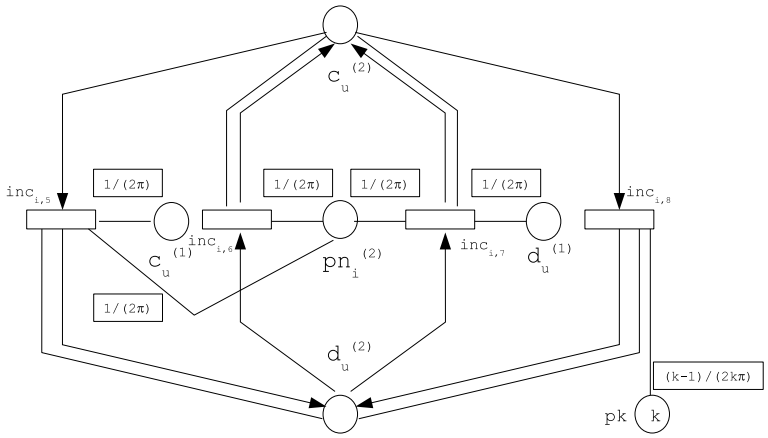

Fig. 21. Incrementing a counter (second stage)

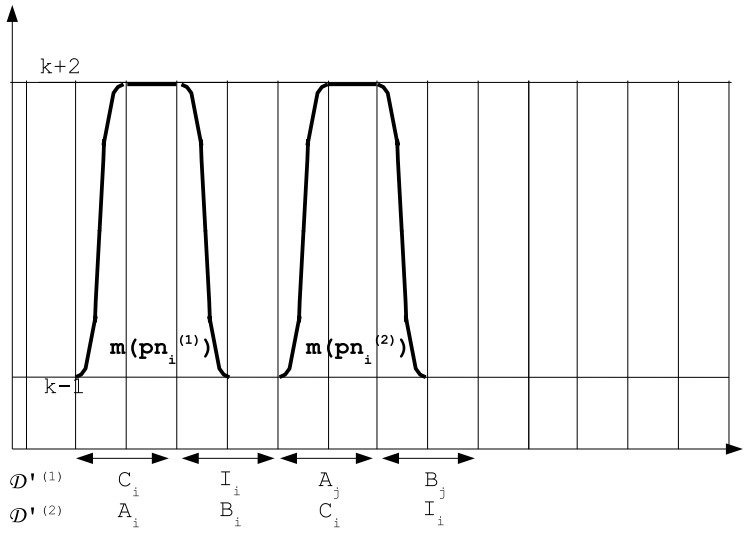

Fig. 22. Non interference between the counter updates

open question whether one can build a simulating bounded net and a simulation mapping with only rational numbers.

4.3 Simulation with a constant number of places

In this paragraph, we show that we can transform our previous simulations such that the number of places is independent from the two counter machine (with $l > 1$ instructions) which is simulated. We develop the case of a robust simulation, the adaptation to a bounded simulation is similar to the one of the previous section.

In order to build such a simulation we change the way, we manage the simulation of the program counter. Instead of places pc_i, pn_i, we will use 4 places:

- pc, qc corresponding to the set of places $\{pc_i\}$,
- pn, qn corresponding to the set of places $\{pn_i\}$,

We do not change places associated with counters. We choose $k \geq 6l^2$.

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 $\mathbf{m}(qc) \in [1, 1 + l/k]$

- $\mathbf{m}(c_1) = k - 1 + 3v_1, \mathbf{m}(c_2) = k - 1 + 3v_2$.

The principle of the simulation is the same as before: a cycle is divided in two parts. We detail the differences between the new simulation and the original one.

First stage: simulation of an unconditional jump 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 23 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 $((k+3l)/(l+1-i))\mathbf{m}(qc) \geq ((k+3l)/(l+1-i))((l+1-i)-l/k) \geq k+2$.

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.

First stage: counter updates The counter updates is performed exactly as in the first simulation (see figure 12).

First stage: simulation of a conditional jump The first stage for simulating the instruction $\mathtt{I_i}$: `if` $\mathtt{cpt_u} = \mathtt{0}$ `then goto` $\mathtt{I_j}$ `else goto` $\mathtt{I_{j'}}$; is illustrated in figure 24 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 is also active (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 desired.

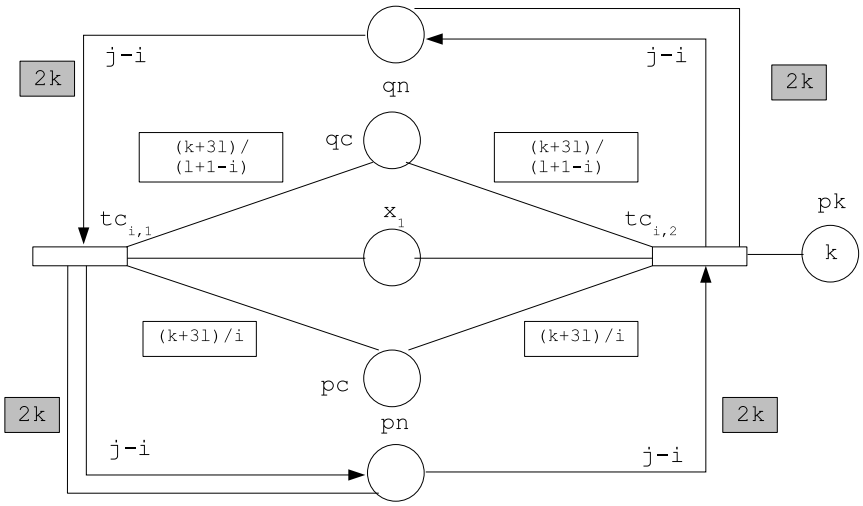

Fig. 23. First stage: simulation of an unconditional jump

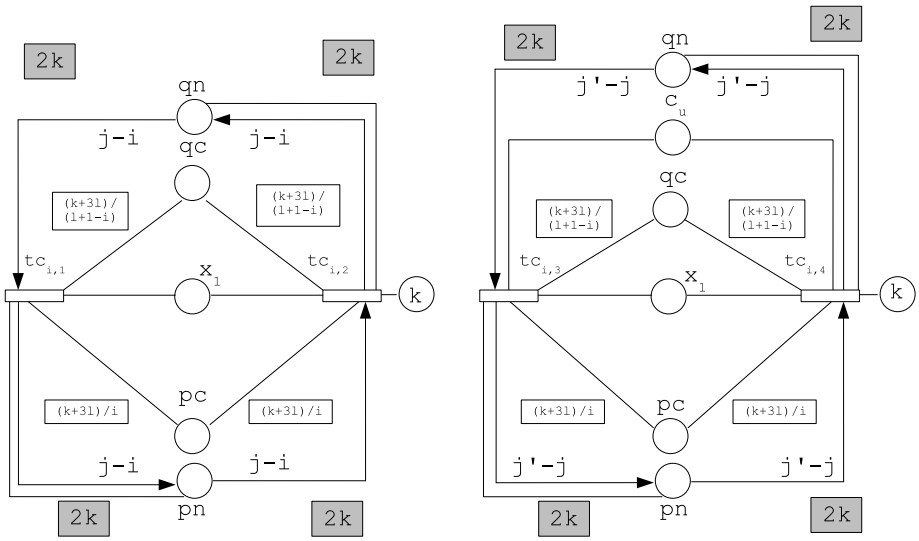

Fig. 24. The first stage of a conditional jump

The second stage The second stage consists in trying to make the marking of pc as close as possible to j and the one of pc 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 25 (the other case is similar).

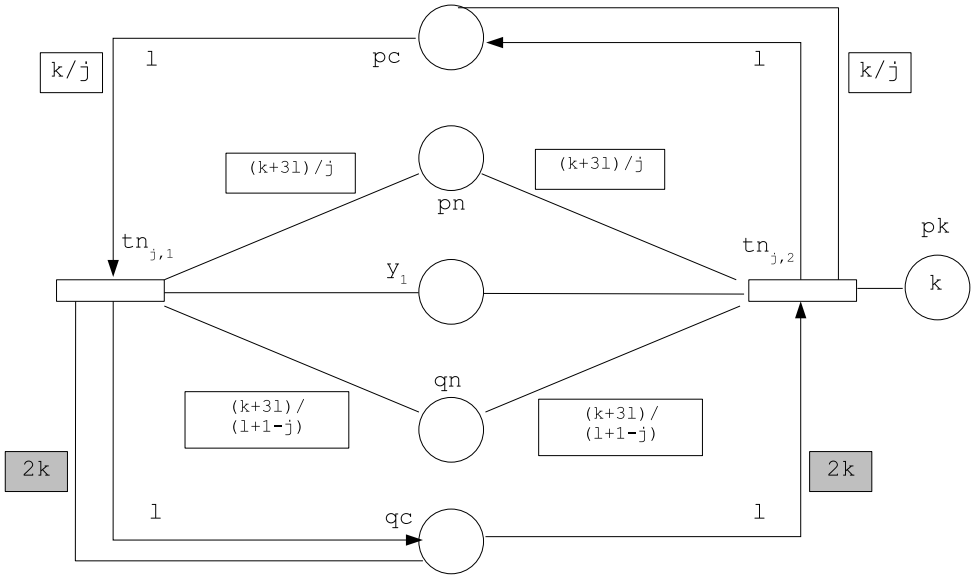

Fig. 25. The second stage

The transition pair $tn_{j,1}, tn_{j,2}$ is activated if both $\mathbf{m}(pn) = j$, $\mathbf{m}(qn) = l+1-j$ 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 which is impossible since $\mathbf{m}(pc) - j \leq l-1$. 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 $l+1-j \leq m(qc) \leq l+1-j-l/k$.

We could count the number of places for the two kinds of simulation. However, it is more appropriate to look at the dimension of the associated ODE. Indeed, places like pk which hold a constant number of tokens yield a constant in the ODE. Furthermore complementary places like qc and pc yield a single dimension since the marking of one is a constant minus the marking of the other.

Summarizing, the clock subnet has dimension 2 (x_1, x_2) and the operating subnet in the unbounded case has dimension 4 (pn, qn, c_1, c_2). In the bounded

case, the operating subnet has dimension 8 (places $c_u^{(s)}$ and $d_u^{(s)}$ are complementary).

Theorem 4. *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 and whose associated ODE has dimension 6 such that $\mathcal{D}$ robustly simulates $\mathcal{M}$,*
- *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 10 such that $\mathcal{D}$ simulates $\mathcal{M}$,*

4.4 Undecidability results

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

Proposition 1 (Coverability and reachability). *Let $\mathcal{D}$ be a (resp. bounded) TDPN whose associated ODE has dimension 6 (resp. 10), $\mathbf{m}_0, \mathbf{m}_1$ 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,*

Proof. We reduce the termination problem of a two counter machine to the three problems. Let $\mathcal{M}$ be a two counter machine and $\mathcal{D}$ the net simulating $\mathcal{M}$. Number the instructions such that $\mathtt{I}_1$, the last instruction, is the stop instruction. Then $\mathcal{M}$ terminates iff $\mathcal{D}$ reaches a marking $\mathbf{m}$ such that $\mathbf{m}(pn) = l$ or equivalently $\mathbf{m}(pn) \geq l$ ($\mathbf{m}(pn^{(1)}) = l$) or $\mathbf{m}(pn^{(1)}) \geq l$) for the bounded simulation). The third problem is a direct consequence of the second one.

Note that we cannot state the undecidability of the reachability problem since in the simulation places *pc* and *qc* are not required to take precise values. However with additional constructions, we show below that the steady-state analysis, a kind of ultimate reachability, is undecidable.

Proposition 2 (Steady-state analysis). *Let $\mathcal{D}$ be a (resp. bounded) TDPN whose associated ODE has dimension 8 (resp. 12), $\mathbf{m}_0$ 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.*

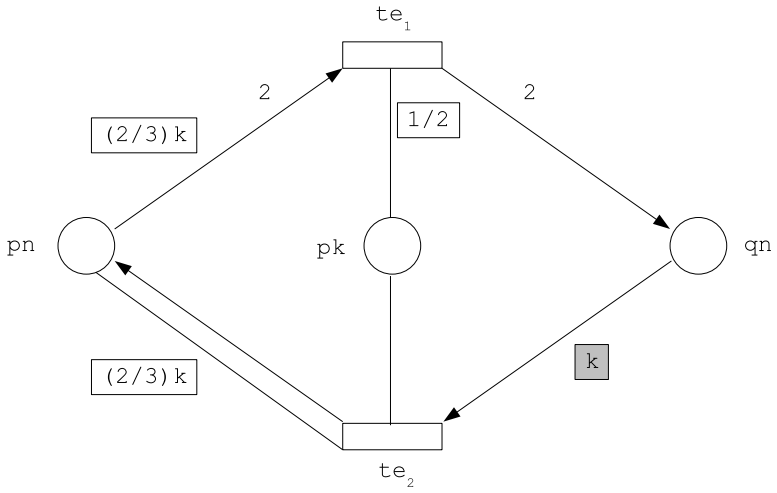

Fig. 26. Ultimate behaviour of pn

Proof. We reduce the termination problem of a two counter machine to this problem. Let $\mathcal{M}$ be a two-counter machine and $\mathcal{D}$ the net simulating $\mathcal{M}$. We assume now that the stop instruction is $\mathtt{I}_1$. We assume also that $\mathtt{I}_1$ can be reached only by $\mathtt{I}_2$, an unconditional jump

At the end of the simulation if the machine stops, it will lead to $\mathbf{m}(pn) = 1$. Taking the beginning of $\mathtt{I}_2$ simulation as the origin of time, we obtain $\mathbf{m}(pn) = 3/2 + 1/2cos(2\tau)$.

We have added transitions te_1 and te_2 to the simulated net. As long as $\mathbf{m}(pn) \geq 3/2$,

- The firing rate of te_1 is determined by pk and equals to $k/2$ thus the consuming rate of tokens in pn is k tokens.
- The firing rate of te_2 is determined by pk and equals to k thus the production rate of tokens in pn is k tokens.

So they mutually cancel their effect.

Assume that the stop instruction is reached then $\mathbf{m}(pn)$ will decrease from 2 to 1. Note that when $\mathbf{m}(pn) = 3/2$ it will still decrease but more quickly due to the fact that the firing rate of te_1 will be greater than the one of te_2. Thus it will reach $5/4$ before $\pi/3$.

Let us write an inequation ruling the behaviour of $\mathbf{m}(pn)$ when it belongs to $[3/4, 5/4]$.

$$\dot{\mathbf{m}}(pn) \leq (2/3)k\mathbf{m}(pn) - k$$

Choosing now the instant when $\mathbf{m}(pn) = 5/4$ as the origin of time, we obtain:

$$\mathbf{m}(pn) \leq 3/2 - 1/4e^{(2/3)k\tau}$$

Since we can choose k as large as required, we may suppose that $m(pn)$ reaches $3/4$ before the end of the first part of the cycle. At this point the inequation becomes:

$$\dot{\mathbf{m}}(pn) \leq -(2/3)k\mathbf{m}(pn)$$

Choosing now the instant when $\mathbf{m}(pn) = 3/4$ as the origin of time, we obtain:

$$\mathbf{m}(pn) \leq (3/4)e^{-(2/3)k\tau}$$

Now, look at the transitions te_3, te_4 of figure 27. They have no effect on the behaviour of the net as long as $m(pn) \geq 3/4$. At the instant when pn reaches $3/4$, we can state the following inequation:

$$\dot{\mathbf{m}}(pc) \leq -k + ke^{-(2/3)k\tau}$$

yielding

$$\mathbf{m}(pc) \leq l + 1 + 3/2 - k\tau - 3/2e^{-(2/3)k\tau}$$

which holds until pc, decreasing, reaches 1. Again we can choose k enough large in order for pc to reach 1 before the end of the first part of the cycle. At this instant, we can state the following equation:

$$\dot{\mathbf{m}}(pc) = -k\mathbf{m}(pc) + (4/3)k\mathbf{m}(pn)$$

which holds as long as $\mathbf{m}(pc) \geq (4/3)\mathbf{m}(pn)$. Let us prove that this always holds. Taking the instant of where pc reaches 1 as the time origin, the resolution of this equation gives:

$$\mathbf{m}(pc)(\tau) = (\mathbf{m}(pc)(0) + \int_0^\tau (4/3)k\mathbf{m}(pn)(s)e^{ks}ds)e^{-k\tau}$$

$$\geq (\mathbf{m}(pc)(0) + (4/3)\mathbf{m}(pn)(\tau)\int_0^\tau ke^{ks}ds)e^{-k\tau}$$

since $(4/3)\mathbf{m}(pn)$ is decreasing.

$$= (\mathbf{m}(pc)(0) + (4/3)\mathbf{m}(pn)(\tau)(e^{k\tau} - 1))e^{-k\tau}$$

$$= (4/3)\mathbf{m}(pn)(\tau) + (\mathbf{m}(pc)(0) - (4/3)\mathbf{m}(pn)(\tau))e^{-k\tau}$$

$$\geq (4/3)\mathbf{m}(pn)(\tau)$$

since $\mathbf{m}(pc)(0) \geq (4/3)\mathbf{m}(pn)(0)$ and $(4/3)\mathbf{m}(pn)$ is decreasing. Thus:

$$\dot{\mathbf{m}}(pc) \leq -km(pc) + ke^{-(2/3)k\tau}$$

So:

$$\mathbf{m}(pc)(\tau) \leq (3e^{(2/3)k\tau} - 2)e^{-k\tau}$$

Since k can be taken enough large, we can suppose that $\mathbf{m}(pc)$ and $\mathbf{m}(pc)$ reach $1/2$ before the end of the cycle. So no transition pairs of the operating subnet

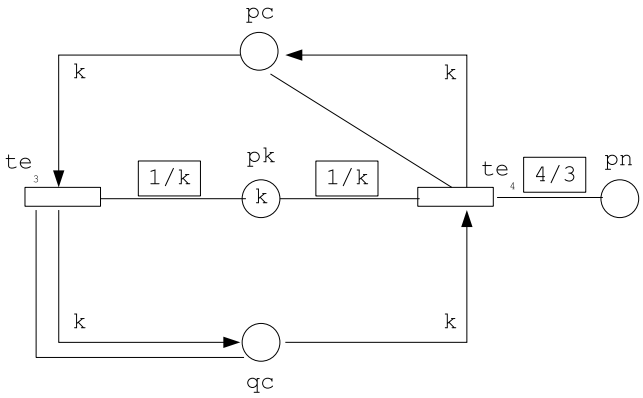

Fig. 27. Emptying *pc*

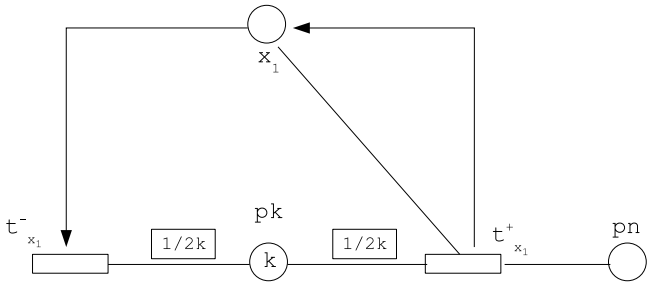

Fig. 28. Emptying x_1

can no more be activated in the future and we conclude that $lim_{\tau\to\infty}\mathbf{m}(pc)(\tau) = lim_{\tau\to\infty}\mathbf{m}(pn)(\tau) = 0$ and $lim_{\tau\to\infty}\mathbf{m}(qc)(\tau) = lim_{\tau\to\infty}\mathbf{m}(qn)(\tau) = l + 1$.

Thus all the places except those related to the clock subnet reach a steady-state. Furthermore, the places simulating the counters reach this state in finite time. We now show how to ultimately empty the places related to the clock subnet. We add to the net a pair of transitions per place of this kind. Figure 28 represents such a net for place x_1. Transition $t^-_{x_1}$ and $t^+_{x_1}$ have opposite incidence and identical firing rates whenever $\mathbf{m}(pn) \geq \inf(\mathbf{m}(x_1), 1/(2k))$. In the general case, the firing rate of $t^-_{x_1}$ is greater or equal than the one of $t^+_{x_1}$. Combining this observation with the structure of the clock subnet, we conclude that $\mathbf{m}(x_1) + \mathbf{m}(y_1)$ decreases and reaches a limit, say v. It remains to show that $v = 0$.
Assume that $v > 0$.
Examining figures 6 and 28, we obtain: $\dot{\mathbf{m}}(x_1) \geq -2k\mathbf{m}(x_1)$. Thus given $\tau \leq \tau'$, one has $\mathbf{m}(x_1)(\tau') \geq \mathbf{m}(x_1)(\tau)e^{-(2k+1)(\tau'-\tau)}$. We note $v' = \inf(v/2, 1/(2k))$. Choose some $0 < \epsilon < \int_0^{\log(2)/(2k+1)}(v'e^{-(2k+1)(s)} - v'/2)ds$ Choose a time instant τ such that both $\mathbf{m}(x_1)(\tau) + \mathbf{m}(y_1)(\tau) \leq v + \epsilon$ and $\mathbf{m}(pn)(\tau) < v'/2$.

We know that either $\mathbf{m}(x_1)(\tau) \geq v'$ or $\mathbf{m}(y_1)(\tau) \geq v'$. Assume that the former case holds (the other one is similar). Then applying the difference between rates of $t^-_{x_1}$ and $t^+_{x_1}$, one obtains:

$$\mathbf{m}(x_1)(\tau + \log(2)/(2k+1)) + \mathbf{m}(y_1)(\tau + \log(2)/(2k+1)) \leq$$

$$\mathbf{m}(x_1)(\tau) + \mathbf{m}(y_1)(\tau) - \int_0^{\log(2)/(2k+1)}(v'e^{-(2k+1)(s)} - v'/2)ds <$$

$$v + \epsilon - \epsilon < v$$

The last inequation is impossible. So $v = 0$.

If $\mathcal{M}$ does not terminate, the additional transitions do not modify the behaviour of the simulating net $\mathcal{D}$ and there is no steady-state due to the presence of the clock subnet.

Finally note that in the modified net, neither $\mathbf{m}(x_1) + \mathbf{m}(y_1)$ nor $\mathbf{m}(x_2) + \mathbf{m}(y_2)$ are linear invariants. Thus the number of dimensions of associated ODEs is increased by two.

5 TDPNs and Timed Continuous Petri Nets (TCPN)

In this section it will be shown that for any TDPN, a TCPN which is equivalent under infinite servers semantics (or variable speed) [7,13] can be obtained, and *vice versa*. Also, it will be seen that in TDPNs the two patterns in the bottom of Figure 1 can be simulated with the two ones on top. In a similar way, TCPN nets with self-loops can be simulated using pure nets (i.e. without self-loops).

Definition 8. *A Timed Continuous Petri Net* $\mathcal{S} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \boldsymbol{\lambda}\rangle$ *is defined by:*

- *P, a finite set of places,*

– T, *a finite set of transitions with* $P \cap T = \emptyset$,
– $\mathbf{Pre}$ *and* $\mathbf{Post}$ *are* $|P| \times |T|$ *sized, natural valued,* incidence matrices,
– $\boldsymbol{\lambda}$, *the speed control vector in* $(\mathbb{R}_{>0})^{|T|}$

As in TDPN, the dynamic of the system is defined by $\dot{\mathbf{m}} = \mathbf{C} \cdot \mathbf{f}(\mathbf{m})$. The difference is in how $\mathbf{f}$ is defined. Under infinite servers semantics (or variable speed) the instantaneous firing rate is defined as:

$$\mathbf{f}(\mathbf{m})(t) = \min(\mathbf{m}(p)/\mathbf{Pre}(p,t) \mid \mathbf{Pre}(p,t) \neq 0) \cdot \boldsymbol{\lambda}(t)$$

The Petri net structure of TCPN is quite similar to that of TDPN, the only difference is that a TDPN is pure regarding the token flow, while a TCPN may not be so, hence we need both the $\mathbf{Pre}$ and $\mathbf{Post}$ incidence matrix, instead of just one $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$. However, they are different as how the marking evolution is defined. In both cases, the minimum operator appears related to the marking, but the places that are used to compute it are slightly different. Moreover, in TDPN the instantaneous firing speed and the effective flow are computed using different elements (the control matrix, $\mathbf{W}$ and the token-flow matrix, $\mathbf{C}$). Hence, control and material flow are separated. However, in TCPN, the pre-incidence matrix, $\mathbf{Pre}$, is used in both, thus a coupling between control and material flow appears. However, it can be seen that in fact they are equivalent in the sense that the dynamical equations obtained with one model can be obtained also using the other, as long as the elements of the speed control matrix are in the rationals. To prove that, observe first that fractions in the $\mathbf{Pre}$ and $\mathbf{Post}$ matrices do not pose a problem, since they can be easily avoided multiplying the columns of the $\mathbf{Pre}$ and $\mathbf{Post}$ matrices by the right number. Observe that this does not change at all the dynamics of the system.

Let $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W} \rangle$ be a TDPN.
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 $\mathbf{W}^*(t) \geq \max(\mathbf{W}(p,t) \cdot (-\mathbf{C}(p,t)) | C(p,t) < 0) > 0$. 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

– $\mathbf{C}'(p,t) = \mathbf{Post}(p,t) - \mathbf{Pre}(p,t)$
– $\mathbf{W}(p,t) = \boldsymbol{\lambda}(t)/\mathbf{Pre}(p,t)$

As an example, the net in Fig. 29 corresponds to applying the previous procedure to the net in Fig 6. Observe that the transformation in both cases does not change the number of places and transitions.

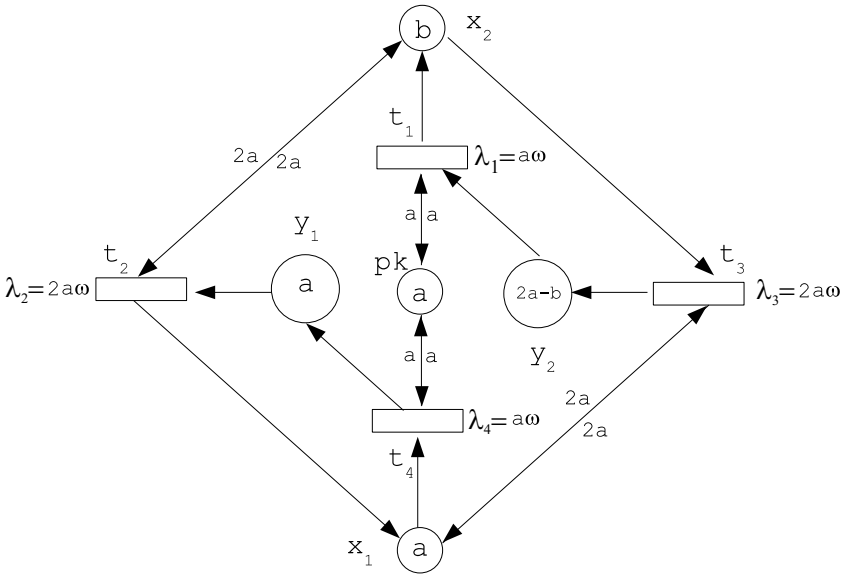

Fig. 29. TCPN equivalent to the TDPN in Fig. 1 (ω being in all transitions defines a time scale)

As claimed in Section 2, the two patterns in the bottom of Fig. 1 can be avoided. We duplicate every place (resp. transition) into a "plus" and a "minus" place (resp. transition). Due to our construction, at any time the places of a pair will have the same marking and the transitions of a pair will have the same firing rate. The plus (resp. minus) places control the speed of the plus (resp. minus) transitions as is done in the original net and there is no other speed control. If a place p and a transition t are connected by a "good" pattern then $\mathbf{C}(p^+, t^+)$ (resp. $\mathbf{C}(p^-, t^-)$) is equal to $\mathbf{C}(p, t)$ and the cross incidences are null. If a place p and a transition t are connected by a "bad" pattern then we define the incidence of the two transitions of the pair in order that their cumulated incidence on the places of the pair are the same as the original one whereas the bad pattern has disappeared.

More formally, let $\mathcal{D} = \langle P, T, \mathbf{C}, \mathbf{W} \rangle$ be a TDPN, then we define an equivalent net without "bad" patterns $\mathcal{D}' = \langle P', T', \mathbf{C}', \mathbf{W}' \rangle$ as follows:

- $P' = \{p^-, p^+ \mid p \in P\}$
- $T' = \{t^-, t^+ \mid t \in T\}$
- If $\mathbf{C}(p, t) < 0 \vee \mathbf{W}(p, t) = \perp$ then
  $\mathbf{C}'(p^-, t^-) = \mathbf{C}'(p^+, t^+) = \mathbf{C}(p, t)$ and $\mathbf{C}'(p^-, t^+) = \mathbf{C}'(p^+, t^-) = 0$
- If $\mathbf{C}(p, t) \geq 0 \wedge \mathbf{W}(p, t) \neq \perp$ 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 $\mathbf{W}'(p^-, t^+) = \mathbf{W}'(p^+, t^-) = \perp$

In practice, we can avoid to duplicate every place and transition depending on the occurrences of the "bad" patterns in the net.

As a first example, let us take a subnet of the net of Fig. 6 presented in the left part of Fig. 30. The only bad pattern is related to x_2 and t_2. We can avoid to duplicate the other places and transitions by an appropriate "re-weighting" of the arcs. The result can be seen in the right part of Fig. 30.

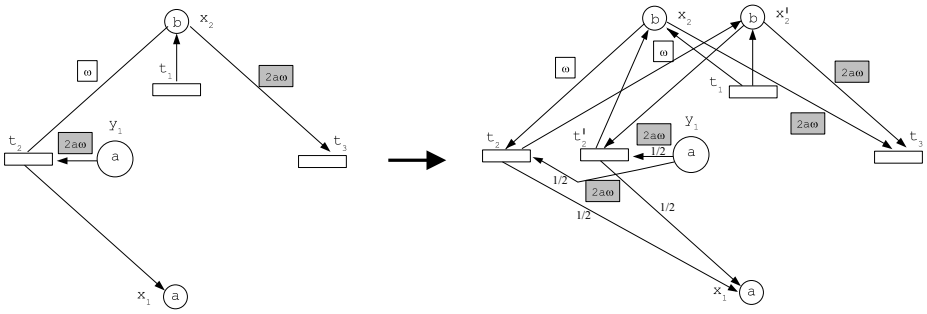

Fig. 30. A simpler local transformation

As a second example, if a place is connected only by control arcs to two transitions, it is enough to put instead a simple circuit with two places with the same number of tokens this place had. For example, this can be used in another subnet of Fig. 6, as shown in Fig. 31.

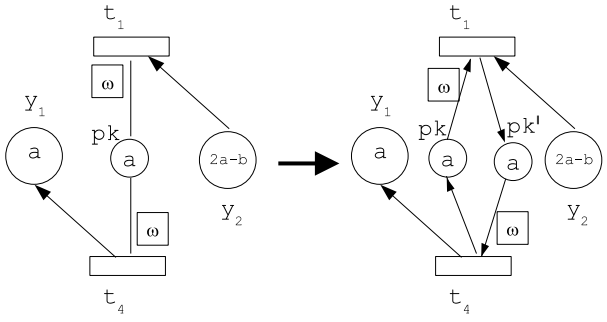

Fig. 31. Another simple transformation to remove control arcs

A similar transformation can be done to *remove self-loop arcs* in TCPN. We detail only the case of one self-loop. We delete the self-loop by transforming it into 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 (two in some cases). Any other input place is input of all the transitions. Any other output transition of the place is now output of all the places. More formally, let $\mathcal{S} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \boldsymbol{\lambda} \rangle$ be the net with the self-loop, and $p \in P$, $t \in T$ be the elements of the self-loop. Let us define the new net $\mathcal{S}' = \langle P', T', \mathbf{Pre}', \mathbf{Post}', \boldsymbol{\lambda}' \rangle$ as follows:

- $P' = P \cup \{p'\}$
- $T' = T \cup \{t', t''\}$
- $\forall t_i \in T \setminus \{t, t', t''\}, \boldsymbol{\lambda}'(t_i) = \boldsymbol{\lambda}(t_i)$. Let $k > 2$ be such that $k \cdot \mathbf{Post}(p,t) - \mathbf{Pre}(p,t) \geq 0$, then $\boldsymbol{\lambda}'(t) = \boldsymbol{\lambda}'(t') = \boldsymbol{\lambda}(t)/k$, and $\boldsymbol{\lambda}'(t'') = \boldsymbol{\lambda}(t) \cdot (1 - 2/k)$
- $\mathbf{Pre}'(p,t) = \mathbf{Pre}'(p,t'') = \mathbf{Pre}'(p',t') = \mathbf{Pre}'(p',t'') = \mathbf{Pre}(p,t)$, and $\mathbf{Post}(p',t) = \mathbf{Post}(p,t') = k \cdot \mathbf{Post}(p,t) - \mathbf{Pre}(p,t)$
- $\forall t_i \in {}^\bullet p, \mathbf{Post}'(p,t_i') = \mathbf{Post}'(p,t_i) = \mathbf{Post}(p,t_i)$ and $\forall t_i \in p^\bullet \setminus \{t\}, \mathbf{Post}'(p,t_i) = \mathbf{Post}'(p',t_i) = \mathbf{Post}(p,t_i)$
- $\forall p_i \in {}^\bullet t \setminus \{p\}, \mathbf{Pre}'(p_i,t'') = \mathbf{Pre}'(p_i,t') = \mathbf{Pre}'(p_i,t) = \mathbf{Pre}(p_i,t)$ and $\forall p_i \in t^\bullet \setminus \{p\}, \mathbf{Post}'(p_i,t'') = \mathbf{Post}'(p_i,t') = \mathbf{Post}'(p_i,t) = \mathbf{Post}(p_i,t)$
- $\forall p_i \in P \setminus \{p\}, \forall t_j \in T \setminus \{t\}, \mathbf{Pre}'(p_i,t_j) = \mathbf{Pre}(p_i,t_j)$ and $\mathbf{Post}'(p_i,t_j) = \mathbf{Post}(p_i,t_j)$

Observe that if $2 \cdot \mathbf{Post}(p,t) - \mathbf{Pre}(p,t) > 0$, then $k = 2$ verifies the restriction. In that case, transition t'' is not really used ($\boldsymbol{\lambda}(t'') = 0$) and can be removed.

The procedure can be easily generalized to the case in which several places are connected with self-loops to the same transition, or when one place is engaged in several self-loops. This means that again this is a linear transformation. The schema in Fig. 32 resumes the relationships between the formalisms and the increase in the size of the model that the simulation may represent.

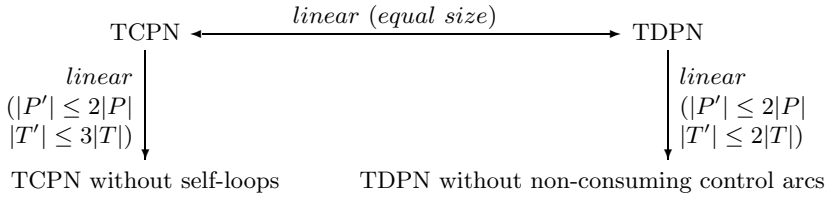

Fig. 32. Relationships between TDPN and TCPN and their versions without self-loops or non-consuming control arcs

Theorem 5. *The expressive power of TDPN, TDPN constrained to the use of the two basic constructions (the two on the top in Fig. 1), TCPN and TCPN without self-loops are identical. Moreover, the transformations range from keeping the places and transitions to a moderate linear increase in size, as can be seen in the diagram in Fig. 32.*

6 Conclusion

In this work, we have introduced a model of differentiable dynamic systems, Time Differentiable Petri Nets, and we have studied their computational power. We emphasize that 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 ODEs $\dot{x} = \mathbf{M} \cdot x$ (with $\mathbf{M}$ a matrix) inside polyhedra.

More precisely, we have designed different simulations of two counter machines in order to fulfill opposite requirements like robustness (allowing some perturbation of the simulation) and boundedness of the simulating net system. Furthermore these simulations can be performed by a net with a constant number of places, i.e. whose dimension of associated ODE is constant. Summarizing our results, TDPNs whose associated ODE is in $(\mathbb{R}_{\geq 0})^6$ can robustly simulate Turing machines and bounded TDPNs whose associated ODE is in $[0, K]^{10}$ (for some fixed K) can simulate Turing machines.

Afterwards, by modifying the simulation, we have proved that marking coverability, submarking reachability and the existence of a steady-state are undecidable.

Finally, we have studied the relations between TDPNs and time continuous Petri nets under infinite server semantics and shown that these models are equivalent in expressiveness and conciseness.

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 TDPNs where the restrictions will be related to both the structure of the net and the associated ODE.

References

1. Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138-1:35–65, 1995.
3. Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, 1998.
4. F. Balduzzi, A. Di Febbraro, A. Giua, and C. Seatzu. Decidability results in first-order hybrid petri nets. *Discrete Event Dynamic Systems*, 11(1 and 2):41–58, 2001.
5. Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems. *Theory of Computing Systems*, 32(1):35–67, 1999.
6. Michael S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 1995.
7. R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets.* Springer-Verlag, 2004.
8. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin 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. Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, 2001.
11. Venkatesh Mysore and Amir Pnueli. Refining the undecidability frontier of hybrid automata. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science 25th International Conference*, volume 3821 of *LNCS*, pages 261–272, Hyderabad, India, 2005. Springer.
12. Hava T. Siegelmann and Eduardo D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360, 1994.
13. M. Silva and L. Recalde. Petri nets and integrality relaxations: A view of continuous Petri nets. *IEEE Trans. on Systems, Man, and Cybernetics*, 32(4):314–327, 2002.