Time Petri Nets with Dynamic Firing Dates: Semantics and Applications

Silvano Dal Zilio, Lukasz Fronc, Bernard Berthomieu, François Vernadat

Time Petri Nets with Dynamic Firing Dates: Semantics and Applications

Silvano Dal Zilio[1,2], Łukasz Fronc[1,2], Bernard Berthomieu[1,2], and François Vernadat[1,3]

[1]CNRS, LAAS, F-31400 Toulouse, France
[2]Univ de Toulouse, LAAS, F-31400 Toulouse, France
[3]Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

We define an extension of time Petri nets such that the time at which a transition can fire, also called its *firing date*, may be dynamically updated. Our extension provides two mechanisms for updating the timing constraints of a net. First, we propose to change the static time interval of a transition each time it is newly enabled; in this case the new time interval is given as a function of the current marking. Next, we allow to update the firing date of a transition when it is persistent, that is when a concurrent transition fires. We show how to carry the widely used state class abstraction to this new kind of time Petri nets and define a class of nets for which the abstraction is exact. We show the usefulness of our approach with two applications: first for scheduling preemptive task, as a poor man's substitute for stopwatch, then to model hybrid systems with non trivial continuous behavior.

1. Introduction

A *Time Petri Net* [15, 6] (TPN) is a Petri net where every transition is associated to a static time interval that restricts the date at which a transition can fire. In this model, time progresses with a common rate in all the transitions that are enabled; then a transition t can fire if it has been continuously enabled for a time θ_t and if the value of θ_t is in the static time interval, denoted $\mathbf{I}_s(t)$. The term static time interval is appropriate in this context. Indeed, the constraint is immutable and do not change during the evolution of the net. In this paper, we lift this simple restriction and go one step further by also updating the timing constraint of persistent transitions, that is transitions that remain enabled while a concurrent transition fires. In a nutshell, we define an extension

of TPN where the time at which a transition can fire, also called its *firing date*, may be dynamically updated. We say that these transitions are fickle and we use the term Dynamic TPN to refer to our extension.

Our extension provides two mechanisms for updating the timing constraints of a net. First, we propose to change the static time interval of a transition each time it is newly enabled. In this case the new time interval $\mathbf{I}_s(t, m)$ is obtained as a function of the current marking m of the net. Likewise, we allow to update the deadline of persistent transitions using an expression of the form $\mathbf{I}_d(t, m, \varphi_t)$, that is based on the previous firing date of t. The first mechanism is straightforward and quite similar to an intrinsic capability of Timed Automata (TA); namely the possibility to compare a given clock to different constants depending on the current state. Surprisingly, it appears that this extension has never been considered in the context of TPN. The second mechanism is far more original. To the best of our knowledge, it has not been studied before in the context of TPN or TA, but there are some similarities with the updatable timed automata of Bouyer et al. [9].

The particularity of timed models, such as TPN, is that state spaces are typically infinite, with finite representations obtained by some abstractions of time. In the case of TPN, states are frequently represented using composite abstract states, or *state classes*, that captures a discrete information (e.g. the marking) together with a timing information (represented by systems of difference constraints or zones). We show how to carry the state class abstraction to our extended model of TPN. We only obtain an over-approximation of the state space in the most general case, but we define a class of nets for which the abstraction is exact. We conjecture that our approach could be used in other formal models for real-time systems, such as timed automata for instance.

There exist several tools for reachability analysis of TPN based on the notion of state class graph [5, 3], like for example Tina [7] or Romeo [14]. Our construction provides a simple method for supporting fickle transitions in these tools. Actually, our extension has been implemented inside the tool Tina in a matter of a few days. We have used this extension of Tina to test the usefulness of our approach in the context of two possible applications: first for scheduling preemptive task, as a poor man's substitute for stopwatch; next to model dynamical systems with non trivial continuous behavior.

Outline of the paper and contributions. We define the semantics of TPN with dynamic firing dates in Sect. 2. We prove that we directly subsume the class of "standard" TPN and that our extension often leads to more concise models. In Sect. 2.3, we motivate our extension by showing how to implement the Quantized State System (QSS) method [10]. This application underlines the advantage of using an asynchronous approach when modeling hybrid systems. Section 3 provides an incremental construction for the state class graph of a dynamic TPN. Before concluding, we give some experimental results for two possible applications of dynamic TPN.

2. Time Petri nets and Fickle Transitions

A *Time Petri net* is a Petri net where transitions are decorated with static time intervals that constrain the time a transition can fire. We denote $\mathbb{I}$ the set of possible time intervals. We use a dense time model in our definitions, meaning that we choose for $\mathbb{I}$ the set of real intervals with non negative rational end-points. To simplify the definitions, we only consider the case of closed intervals, $[a, b]$, and infinite intervals of the form $[a, +\infty)$. For any interval i in $\mathbb{I}$, we use the notation $\downarrow i$ for its left end-point and $\uparrow i$ for its right end-point (or ∞ if i is unbounded).

We use the expression Dynamic TPN (DTPN) when it is necessary to make the distinction between our model and more traditional definitions of TPN. With our notations, a dynamic TPN is a tuple $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0, \mathbf{I}_s, \mathbf{I}_d \rangle$ in which:

- $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0 \rangle$ is a Petri net, with P the set of places, T the set of transitions, $m_0 : P \to \mathbb{N}$ the initial marking, and $\mathbf{Pre},\ \mathbf{Post} : T \to P \to \mathbb{N}$ the precondition and postcondition functions.
- $\mathbf{I}_s$ is the *static interval function*, that associates a time interval (in $\mathbb{I}$) to every transition (in T).
- $\mathbf{I}_d$ is the *dynamic interval function*. It will be used to update the firing date of persistent transitions.

We slightly extend the "traditional" model of TPN and allow to define the static time interval of a transition as a function of the markings, meaning that $\mathbf{I}_s$ is a function of $T \to (P \to \mathbb{N}) \to \mathbb{I}$. We will often used the curryied function $\mathbf{I}_s(t)$ to denote the mapping from a marking m to the time interval $\mathbf{I}_s(t, m)$.

We also add the notion of *dynamic interval function*, $\mathbf{I}_d$, that is used to update the firing date of persistent transitions. The idea is to update the firing date φ_t of a persistent transition t using a function of φ_t. Hence $\mathbf{I}_d$ is a function of $T \to (P \to \mathbb{N}) \to \mathbb{R}_{\geq 0} \to \mathbb{I}$. For example, a transition t such that $\mathbf{I}_d(t, m, \theta) = [\theta + 1, \theta + 2]$, for all $\theta \geq 0$, models an event that is delayed by between 1 and 2 unit of time (u.t.) when a concurrent transition fires.

2.1. A Semantics for Time Petri Nets Based on Firing Functions

As usual, we define a *marking* m of a TPN as a function $m : P \to \mathbb{N}$ from places to integers. A transition $t \in T$ is *enabled* at m if and only if $m \geq \mathbf{Pre}(t)$ (we use the pointwise comparison between functions). We denote $\mathcal{E}(m)$ the set of transitions enabled at m.

A *state* of a TPN is a pair $s = (m, \varphi)$ in which m is a marking and $\varphi : T \to \mathbb{R}_{\geq 0}$ is a mapping, called the *firing function* of s, that associates a firing date to every transition enabled at m. Intuitively, if t is enabled at m, then φ_t is the date (in the future, from now) at which t should fire. Also, the transitions that may fire from a state (m, φ) are exactly the transitions t in $\mathcal{E}(m)$ such that φ_t is minimal; they are the first scheduled to fire.

For any date θ in $\mathbb{R}_{\geq 0}$, we denote $\varphi \dot{-} \theta$ the partial function that associates the transition t to the value $\varphi_t - \theta$, when $\varphi_t \geq \theta$, and that is undefined elsewhere. This operation is useful to model the effect of time passage on the enabled transitions of a net. We say that the firing function $\varphi \dot{-} \theta$ is well-defined if it is defined on exactly the same transitions than φ.

The following definitions are quite standard. The semantics of a TPN is a Kripke structure $\langle S, S_0, \to \rangle$ with only two possible kind of actions: either $s \xrightarrow{t} s'$ (meaning that the transition $t \in T$ is fired from s); or $s \xrightarrow{\theta} s'$, with $\theta \in \mathbb{R}_{\geq 0}$ (meaning that we let time θ elapse from s). A transition t may fire from the state (m, φ) if t is enabled at m and firable instantly (that is $\varphi_t = 0$). In a state transition $(m, \varphi) \xrightarrow{t} (m', \varphi')$, we say that a transition k is *persistent* (with $k \neq t$) if it is also enabled in the marking $m - \mathbf{Pre}(t)$, that is if $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$. The transitions that are enabled at m' and not at m are called *newly enabled*. We define the predicates prs and nbl that describe the set of persistent and newly enabled transitions after t fires from m:

$$\begin{array}{rcl} \mathrm{prs}(m,t) & = & \{k \in \mathcal{E}(m) \mid m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)\} \\ \mathrm{nbl}(m,t) & = & \{k \in (T \setminus \mathcal{E}(m)) \cup \{t\} \mid m - \mathbf{Pre}(t) + \mathbf{Post}(t) \geq \mathbf{Pre}(k)\} \end{array}$$

We use these two predicates to define the semantics of DTPN.

Definition 1. *The semantics of a DTPN $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0, \boldsymbol{I}_s, \boldsymbol{I}_d \rangle$ is the timed transition system $SG = \langle S, S_0, \to \rangle$ such that:*

- *S is the set of states of the TPN;*
- *S_0, the set of initial states, is the subset of states of the form (m_0, φ), where m_0 is the initial marking and $\varphi_t \in \boldsymbol{I}_s(t, m_0)$ for every t in $\mathcal{E}(m_0)$;*
- *the state transition relation $\to \subseteq S \times (T \cup \mathbb{R}_{\geq 0}) \times S$ is the smallest relation such that for all state (m, φ) in S:*
  - *(i) if t is enabled at m and $\varphi_t = 0$ then $(m, \varphi) \xrightarrow{t} (m', \varphi')$ where $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$ and φ' is a firing function such that $\varphi'_k \in \boldsymbol{I}_d(k, m', \varphi_k)$ for all persistent transition $k \in \mathrm{prs}(m, t)$ and $\varphi'_k \in \boldsymbol{I}_s(k, m')$ otherwise.*
  - *(ii) if $\varphi \dot{-} \theta$ is well-defined then $(m, \varphi) \xrightarrow{\theta} (m, \varphi \dot{-} \theta)$.*

The state transitions labelled over T (case (i) above) are the *discrete* transitions, those labelled over $\mathbb{R}_{\geq 0}$ (case (ii)) are the *continuous*, or time elapsing, transitions. It is clear from Definition 1 that, in a discrete transition $(m, \varphi) \xrightarrow{t} (m', \varphi')$, the transitions enabled at m' are exactly $\mathrm{prs}(m, t) \cup \mathrm{nbl}(m, t)$. In the target state (m', φ'), a newly enabled transition k get assigned a firing date picked "at random" in $\mathbf{I}_s(k, m')$. Similarly, a persistent transition k get assigned a firing date in $\mathbf{I}_d(k, m', \varphi_k)$. Because there may be an infinite number of transitions, the state spaces of TPN are generally infinite, even when the net is bounded. This is why we introduce an abstraction of the semantics in Sect. 3.

We can define two simple extensions to DTPN. First, we can use a special treatment for re-initialized transitions; transitions that are enabled before t fires and newly-enabled after. In this case we could use the previous firing date to compute the static interval. Then, in the interval functions $\mathbf{I}_d$ and $\mathbf{I}_s$, we can use the "identifier" of the transition that fires in addition to the target marking, m'. These extensions preserve the results described in this paper

Our definitions differ significantly from the semantics of TPN generally used in the literature. For instance, in the works of Berthomieu et al. [1], states are either based on *clocks*—that is on the time elapsed since a transition was enabled—or on *firing domains* (also called time zones)—that abstract the sets of possible "time to fire" using intervals. Our choice is quite close to the TPN semantics based on firing domains (in particular we have the same set of traces) and is similar in spirit to the semantics used by Vicario et al. [18] for reasoning about Stochastic Time Petri nets. We made the choice of an unorthodox semantics to simplify our definition of firing date. We conjecture that most of our definitions can be transposed to a clock-based semantics.

2.2. Interesting Classes of DTPN

In the standard semantics of TPN [15], the firing date of a persistent transition is left unchanged. We can obtain a similar behavior by choosing for $\mathbf{I}_d(t, m, \theta)$ the time interval $[\theta, \theta]$. We say in this case that the dynamic interval function is *trivial*. Another difference with respect to the standard definition of TPN is the fact that the (static!) time interval of a transition may change. We say that a dynamic net is a TPN if its static function, $\mathbf{I}_s$, is constant and its dynamic function, $\mathbf{I}_d$, is trivial. We say that a DTPN is *weak* if only the function $\mathbf{I}_d$ is trivial. We show that TPN are as expressive than weak DTPN when the nets are bounded. Weak nets are still interesting though, since the use of non-constant interval functions can lead to more concise models. On the other hand, the results of Sect. 3 show that, even in bounded nets, fickle transitions are more expressive than weak ones.

Theorem 1. *For every weak DTPN that has a finite set of reachable markings, there is a TPN that has an equivalent semantics.*

Proof. see Appendix A. □

We define a third class of nets, called *translation DTPN*, obtained by restricting the dynamic interval function $\mathbf{I}_d$. This class arises naturally during the definition of the State Class Graph construction in Sect. 3. Intuitively, with this restriction, a persistent transition can only shift its firing date by a "constant time". The constant can be negative and may be a function of the marking. More precisely, we say that a DTPN is a translation if, for every transitions t, there are two functions κ_1 and κ_2 from $(P \to \mathbb{N}) \to \mathbb{Q}$ such that $\mathbf{I}_d(t, m, \theta)$ is the time interval $[A, B]$ where $A = \max(0, \theta + \kappa_1(m))$ and $B = \max(A, \theta + \kappa_2(m))$. (The use of max in the definition of A, B is necessary to accomodate negative constants $\kappa_i(m)$.)

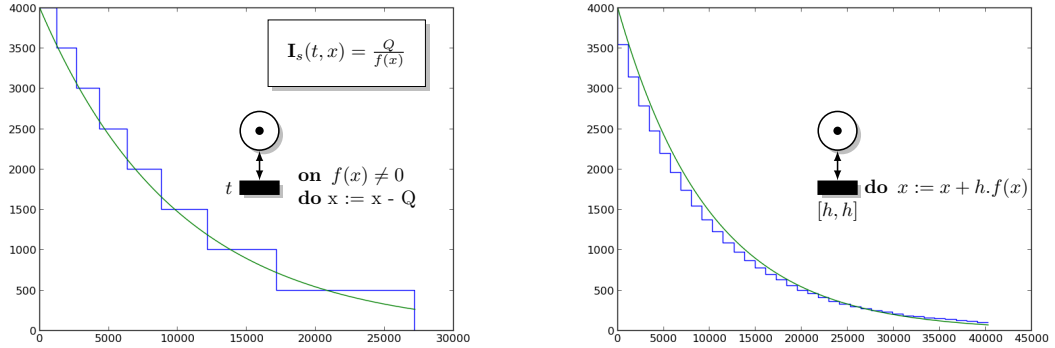


Figure 1: A simple QSS simulation (left) and the Euler method (right) for $\dot{x} = -x$. ($Q = 500$, $h = 1150$, global error smaller than 500.)

2.3. Interpretation of the Quantized State System Model

With the addition of fickle transitions, it is possible to model systems where the timing constraints of an event depend on the current state. This kind of situations arises naturally in practice. For instance, we can use the function $\mathbf{I}_s$ to model the fact that the duration of a communication depends on the length of a message. Likewise, we can use the fickle function $\mathbf{I}_d$ when modeling the typical workflow of a conference, in which a deadline may be postponed when particular events occurs.

In this section, we consider a simple method for analyzing the behavior of a system with one continuous variable, x, governed by the ordinary differential equation $\dot{x} = f(x)$. The idea is to define a TPN that computes the value $x(\theta)$ of the variable x at the date θ. To this end, we use an extension of TPN with shared variables, $x, y, \ldots$, where every transition may be guarded by a boolean predicate (**on** b) and such that, upon firing, a transition can update the environment (using a sequence of assignments, **do** e). This extension of TPN with shared variables can already be analyzed using the tool Tina.

The simplest solution is based on the Euler forward method. This is modeled by the TPN of Fig. 1 (right) that periodically execute the instruction $x := x + h.f(x)$ every h (the value of the time step, h, is the only parameter of the method). This solution is a typical example of synchronous system, where we sample the evolution of time using a "quantum of time". A synchronous approach answers the following question: given the value of x at time $k.h$, what is its value at time $(k+1).h$?

The second solution is based on the Quantized State System (QSS) method [10, 11], which can be interpreted as the dual—the asynchronous counterpart—of the Euler method. QSS uses a "quantum of value", Q, meaning that we only consider discrete values for x, of the form $k.Q$ with $k \in \mathbb{N}$. The idea is to compute the time necessary for x to change by an amount of Q. To paraphrase [11], the QSS method answers the following modified question: given that x has value $k.Q$, what is the earliest time at which x has value $(k \pm 1).Q$? This method has a direct implementation using fickle transitions: at first approximation, the time φ_t for x to change by an amount of Q is given by the

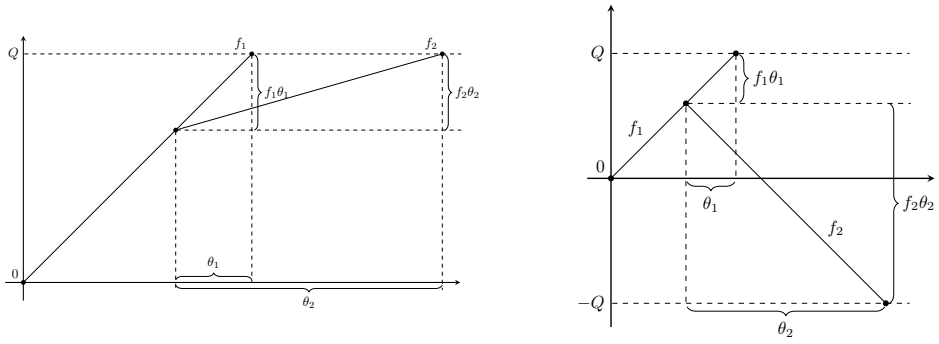


Figure 2: Computing the updated firing date in the QSS method.

relation $(x \pm Q) = x + \varphi_t.f(x)$, that is $\varphi_t = {}^{Q}/_{|f(x)|}$. We have that the *time slope* of x is equal to ${}^{1}/_{f(x)}$. The role of the guard $f(x) \neq 0$ on transition t is to avoid pathological values for the slope; when $f(x)$ is nil the value of x stays constant, as needed.

We compare the results obtained with these two different solutions in Fig. 1, where we choose $f(x) = -x$ and $x(0) = 4000$. Each plot displays the evolution of the TPN compared to the analytic solution, in this case $x(\theta) = 4000e^{-\theta}$. Numerical methods are of course approximate; in both cases (Euler and QSS) the global error is proportional to the quantum. The plots are obtained with the largest quantum values giving a global error smaller than 500, that is a step h of 1150 and a quantum Q of 500. The dynamic TPN has 10 states while the standard TPN has 38. The ratio improves when we try to decrease the global error. For instance, for an error smaller than 100 (which gives $Q = 100$ and $h = 250$) we have 42 states against 182. We observe that in this case the "asynchronous" solution is more concise than the synchronous one.

The Euler method is the simplest example in a large family of iterative methods for approximating the solutions of differential equations. The QSS method used in this section can be enhanced in just the same way, leading to more precise solutions, with better numerical stability. Some of the improved QSS methods have been implemented in our tool, but we still experiment the effect of numerical instability on some stiff systems. In these cases, the synchronous approach (that is deterministic) may sometimes exhibit better performances.

Although we make no use of the fickle function $\mathbf{I}_d$ here, it arises naturally when the system has multiple variables. Consider a system with two variables, x, y, such that $\dot{x} = f(x, y)$. We can use the same solution than in Fig. 1 to model the evolution of x and y. When the value of x just changes, the next update is scheduled at the date ${}^{Q}/_{f(x,y)}$ (the time slope is $f_1 = {}^{1}/_{f(x,y)}$). If the value of y is incremented before this deadline—say that the remaining time if θ_1—we need to update the time slope and use the new value $f_2 = {}^{1}/_{f(x,y+Q)}$.

We illustrate the situation in the two diagrams of Fig. 2, where we assume that f_1 is positive. For instance, if the two slopes have the same sign (diagram to the left), we need to update the firing date to the value θ_2 such that $|f_1|.\theta_1 = |f_2|.\theta_2$. Likewise, when f_2 is negative, we have the relation $|f_1|.\theta_1 + |f_2|.\theta_2 = 2.Q$. Therefore, depending on the sign of $f_1.f_2$ (the sign of $\dot{y}$ tell us whether y is incremented or decremented) we have

$\mathbf{I}_d(t,x,y,\theta) = [A(\theta), A(\theta)]$ with:

$$A(\theta) = \frac{|f(x,y \pm Q)|}{|f(x,y)|}.\theta \qquad \text{or} \qquad \frac{|f(x,y \pm Q)|}{|f(x,y)|}.(2.Q.|f(x,y)| - \theta)$$

This example shows that it is possible to implement the QSS method using only linear fickle functions. (We discuss briefly the associated class of DTPN at the end of Sect. 3.) Since the notion of slope is central in our implementation of the QSS method, we could have used instead an extension of TPN with multirate transitions [12], that is a model where time advance at different rate depending on the state. While the case $f_1.f_2 > 0$ lends itself well to this extension, it is not so obvious when the slopes have different signs. On the opposite, it would be interesting to use fickle transitions as a way to mimic multirate transitions.

3. A State Class Abstraction for Dynamic TPN

In this section, we generalize the state class abstraction method to the case of DTPN. A State Class Graph (SCG) is a finite abstraction of the timed transition system of a net that preserves the markings and traces. The construction is based on the idea that temporal information in states (the firing functions) can be conveniently represented using systems of difference constraints [17]. We show that the SCG faithfully abstract the semantics of a net when the dynamic interval functions are translations. We only over-approximate the set of reachable markings in the most general case.

A state class C is defined by a pair (m, D), where m is a marking and the firing domain D is described by a (finite) system of linear inequalities. We say that two state classes $C = (m, D)$ and $C' = (m', D')$ are equal, denoted $C \cong C'$, if $m = m'$ and $D \Leftrightarrow D'$ (i.e. D and D' have equal solution sets). Hence class equivalence is decidable. In a domain D, we use variables $x_t, y_t, \ldots$ to denote a constraint on the value of φ_t. A domain D is defined by a set of difference constraints in reduced form: $\alpha_i \leq x_i \leq \beta_i$ and $x_i - x_j \leq \gamma_{i,j}$, where i, j range over a given subset of "enabled transitions" and the coefficients α, β and γ are rational numbers. We can improve the reduced form of D by choosing the tightest possible bounds that do not change its associated solutions set. In this case we say that D is in closure form. We show in Th. 2 how to compute the coefficients of the closure form incrementally.

In the remainder of this section, we use the notation $A^s_t(m)$ and $B^s_t(m)$ for the left and right endpoints of $\mathbf{I}_s(t,m)$. Likewise, when the marking m is obvious from the context, we use the notations $A_t(\theta)$ and $B_t(\theta)$ for the left and right endpoints of $\mathbf{I}_d(t,m,\theta)$, that is $A_t(\theta) = \downarrow\mathbf{I}_d(t,m,\theta)$ and $B_t(\theta) = \uparrow\mathbf{I}_d(t,m,\theta)$. We call A_t and B_t the fickle functions of t. In the remainder of the text, we assume that $0 \leq A_t(\theta) \leq B_t(\theta)$ for all possible (positive) date θ and that $A(\infty) = \infty$. We also require these functions to be monotonically increasing. We impose no other restrictions on the fickle functions.

We define inductively a set of classes C_σ, where $\sigma \in T^*$ is a sequence of discrete transitions firable from the initial state. This is the State Class Graph construction of [5, 3]. Intuitively, the class $C_\sigma = (m, D_\sigma)$ "collects" the states reachable from the

initial state by firing schedules of support sequence σ. The initial class C_ϵ is (m_0, D_0) where D_0 is the domain defined by the set of inequalities $A_i^s(m_0) \leq x_i \leq B_i^s(m_0)$ for all i in $\mathcal{E}(m_0)$.

Assume $C_\sigma = (m, D)$ is defined and that t is enabled at m. We details how to compute the domain for the class $C_{\sigma.t}$. First we test whether the system D extended with the constraints $D_t = \{x_k - x_t \geq 0 \mid t \neq k, k \in \mathcal{E}(m)\}$ is consistent. This is in order to check that transition t can be fired before any other enabled transitions k at m. If $D \wedge D_t$ is consistent, we add $C_{\sigma.t} = (m', D')$ to the set of reachable classes, where m' is the result of firing t from m, i.e. $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$. The computation of D' follows the same logic than with standard TPN.

We choose a set of fresh variables, say y_k, for every transition k that is enabled at m'. For every persistent transition, $k \in \mathrm{prs}(m, t)$, we add the constraints $x_k = y_k - x_t$ to the set of inequalities in $D \wedge D_t$. The variable y_k matches the firing date of k at the time t fires, that is, the value of φ_k used in the expression $\mathbf{I}_d(k, m', \varphi_k)$ (see Definition 1, case (i)). For every newly enabled transition, $k \in \mathrm{nbl}(m, t)$, we add the constraints $A_k^s(m') \leq y_k \leq B_k^s(m')$. This constraint matches the fact that φ'_k is in the interval $\mathbf{I}_s(k, m')$ if k is newly enabled at m'. As a result, we obtain a set of inequations where we can eliminate all occurrences of the variables x_k and x_t. After removing redundant inequalities and simplifying the constraints on transitions in conflicts with t—so that the variables only ranges over transitions enabled at m'—we obtain an "intermediate" domain D_{int} that obeys the constraints: $\kappa_i \leq y_i \leq \lambda_i$ and $y_i - y_j \leq \mu_{i,j}$, where i, j range over $\mathcal{E}(m')$ and the constants κ, λ and μ are defined as follows.

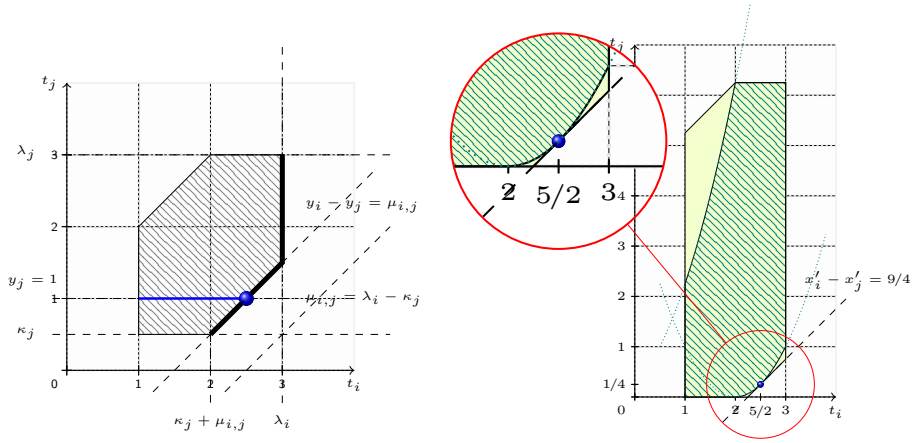
$$
\begin{array}{rcll}
\kappa_i & = & \left\{ \begin{array}{ll} A_i^s(m') & \text{if } i \text{ is newly enabled,} \\ \max\left(0, \{-\gamma_{i,j} \mid i, j \in \mathcal{E}(M)\}\right) & \text{otherwise} \end{array} \right. & \\
\lambda_i & = & \left\{ \begin{array}{ll} B_i^s(m') & \text{if } i \text{ is newly enabled,} \\ \gamma_{i,t} & \text{otherwise} \end{array} \right. & \text{(C1)} \\
\mu_{i,j} & = & \left\{ \begin{array}{ll} \lambda_i - \kappa_j & \text{if either } i \text{ or } j \text{ newly enabled,} \\ \min\left(\gamma_{i,j}, \lambda_i - \kappa_j\right) & \text{otherwise} \end{array} \right. &
\end{array}
$$

Finally, we need to apply the effect of the fickle functions. For this, we rely on the fact that A_i and B_i are monotonically increasing functions. To obtain D', we choose a set of fresh variables, say x'_i, for every transition $i \in \mathcal{E}(m')$ and add the following relations to D_{int}. To simplify the notation, we assume that in the case of a newly enabled transition, j, the functions A_j and B_j stand for the identity function (with this shorthand, we avoid to distinguish cases where both or only one of the transitions are persistent):

$$
\begin{array}{ll}
x'_i = y_i & \text{if } i, j \text{ are newly enabled} \\
A_i(y_i) \leq x'_i \leq B_i(y_i) \text{ and } x'_i - x'_j \leq B_i(y_i) - A_j(y_j) & \text{if } i \text{ or } j \text{ are persistent}
\end{array}
$$

The relation for newly enabled transitions simply states that y_i already captures all the constraints on the firing time φ'_i. For persistent transitions, the first relation states that x'_i is in the interval $[A_i(y_i), B_i(y_i)]$, that is in $\mathbf{I}_d(i, m', \varphi(i))$.

We obtain the domain D' by eliminating all the variables of the kind y_i. First, we can observe that, by monotonicity of the functions A_i and B_i, we have $A_i(\kappa_i) \leq A_i(y_i)$



(a) Domain D_{int} projected over t_i, t_j (b) Domain D' obtained from D_{int}

Figure 3: Computing the coefficient $\gamma'_{i,j}$ in the domain D'.

and $B_i(y_i) \leq B_i(\lambda_i)$. This gives directly a value for the coefficients α'_i and β'_i. The computation of the coefficient $\gamma'_{i,j}$ is more complex, since it amounts to computing the maximum of a function over a convex sets of points. Indeed $\gamma'_{i,j}$ is the least upper-bound for the values of $x'_i - x'_j$ over D_{int} or, equivalently:

$$\begin{array}{rcl} \gamma'_{i,j} & = & \max\{B_i(y_i) - A_j(y_j) \mid y_i, y_j \in D_{int}\} \\ & = & \max\{B_i(y_i) - A_j(y_j) \mid \kappa_i \leq y_i \leq \lambda_i, \kappa_j \leq y_j \leq \lambda_j, y_i - y_j \leq \mu_{i,j}\} \end{array}$$

It is possible to simplify the definition of $\gamma'_{i,j}$. Indeed, if we fix the value of y_j then, by monotonicity of B_i, the maximal value of $B_i(y_i) - A_j(y_j)$ is reached when y_i is maximal. Hence we have two possible cases: either (i) it is reached for $y_i = y_j + \mu_{i,j}$ if $\kappa_j \leq y_j \leq \lambda_i + \mu_{i,j}$; or (ii) it is reached for $y_i = \lambda_i$ if $\lambda_i - \mu_{i,j} \leq y_j \leq \lambda_j$. This result is illustrated in the schema of Fig. 3a, where we display an example of domain D_{int}. When y_j is constant (horizontal line), the maximal value is on the "right" border of the convex set (bold line). We also observe that in case (ii), by monotonicity of A_j, the maximal value is equal to $B_i(\lambda_i) - A_j(\lambda_i + \mu_{i,j})$. Therefore the value of $\gamma'_{i,j}$is obtained by computing the maximal value of the expression $B_i(\theta) - A_j(\theta - \mu_{i,j})$, that is:

$$\gamma'_{i,j} = \max\{B_i(\theta) - A_j(\theta - \mu_{i,j}) \mid \kappa_j + \mu_{i,j} \leq \theta \leq \lambda_i\} \qquad \text{(C2)}$$

As a consequence, the value of $\gamma'_{i,j}$ can be computed by finding the minimum of a numerical function (of one parameter) over a real interval, which is easy.

We display in Fig. 3b the domain D' obtained from D_{int} after applying the fickle functions. In this example, t_j is the only fickle transition and we choose $A_j(\theta) = B_j(\theta) = (\theta - 1/2)^2$ when $\theta \geq 1/2$. With our method we have that $\mu_{i,j} = 3/2$ and the value of $\gamma'_{i,j}$ is obtained by computing the maximal value of the expression $(\theta - 1/2)^2 - (\theta - 3/2)$ with $\theta \in [2, 3]$, that is $9/4$.

Theorem 2. *Assume $C = (m, D)$ is a class with D in closure form. Then for every transition t in $\mathcal{E}(m)$ there is a unique class (m', D') obtained from C by firing t. The domain D' is also in closure form and can be computed incrementally as follows (we assume that A_i and B_i stands for the identity functions when i is newly enabled).*

$$\begin{array}{lcll}
\alpha'_i & = & A^s_i(m') & \textit{if } i \textit{ is newly enabled,} \\
 & = & \max\left(A_i(0), \{A_i(-\gamma_{i,j}) \mid i,j \in \mathcal{E}(M)\}\right) & \textit{otherwise} \\
\beta'_i & = & B^s_i(m') & \textit{if } i \textit{ is newly enabled,} \\
 & = & B_i(\gamma_{i,t}) & \textit{otherwise} \\
\gamma'_{i,j} & = & \min(\gamma_{i,j}, \beta'_i - \alpha'_j) & \textit{if } i,j \textit{ are newly enabled,} \\
 & = & \max\{B_i(\theta) - A_j(\theta - \mu_{i,j}) \mid \mu_{i,j} + \kappa_j \leq x \leq \lambda_i\} & \textit{otherwise} \\
 & & & (\textit{where } \lambda_i, \kappa_j \textit{ and } \mu_{i,j} \textit{ are defined as in (C1)})
\end{array}$$

Moreover, if the state (m, φ) is reachable in the state graph of a net, say N, and $(m, \varphi) \xrightarrow{\theta} \xrightarrow{t} (m', \varphi')$ then there is a class $C_\sigma = (m, D)$ reachable in the SCG computed for N with $\varphi \in D$, $C_{\sigma.t} = (m', D')$ and $\varphi' \in D'$.

The hatched area inside the domain displayed in Fig. 3b is the image of the domain D_{int} after its transformation by the fickle function $A_j(\theta)$. We see that some points of D' have no corresponding states in D_{int}. Hence we only have an over-approximation. (We do not have enough place to give an example of net with a marking that is in reachable in the *SCG* but not reachable in the state space, but such an example is quite easy to build.) If we consider the definition of the coefficients γ' in equation (C2), we observe that the situation is much simpler if the fickle functions are translations. Actually, it is possible to prove that, in this case, the *SCG* construction is exact.

Theorem 3. *If the DTPN N is a translation then the SCG defined in Th. 2 has the same set of reachable markings and the same set of traces than the timed transition system of N.*

sketch. By equation (C2), if the net is a translation then there are two constants c_i, c_j such that $B_i(\theta) = \theta + c_i$ and $A_j(\theta) = \theta + c_j$. Therefore the expression $B_i(\theta) - A_j(\theta - \mu_{i,j})$ is constant and equal to $c_i - c_j - \mu_{i,j}$ (the maximum is reached all over the boundary of the domain). In this case, every state in D' has a corresponding states in D_{int}. □

We can also observe that, if the dynamic interval bounds A_i and B_i are linear functions, then we can follow a similar construct using (general) systems of inequations for the domains instead of difference constraints. This solution gives also an exact abstraction for the state space but is not interesting from a computational point of view (since we loose the ability to compute a canonical form for the domain incrementally). In this case, we are in a situation comparable to the addition of stopwatch to TPN where systems of difference constraints are not enough to precisely capture state classes. With our computation of the coefficient γ', see equation (C2), we use instead the "best difference bound matrix" that contains the states reachable from the class C. This approximation is used in some tools that support stopwatches, like Romeo [14].

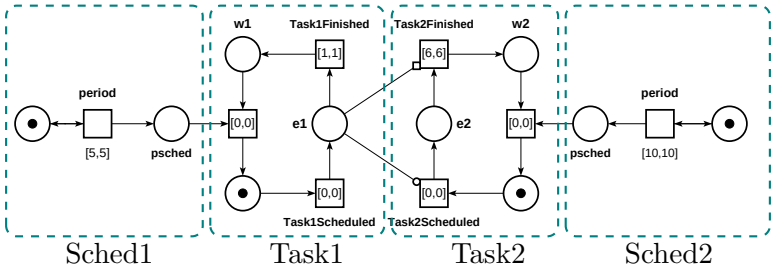


Figure 4: System with one preemptive and one simple task.

4. Two Application for Dynamic TPN

We study two possible applications for fickle transitions. First to model a system of preemptive, periodic tasks with fixed duration. Next to model hybrid system with non trivial continuous behavior. These experiments have been carried out using a prototype extension of Tina. The tool and the all the models are available online at `http://projects.laas.fr/tina/fickle/`.

4.1. Scheduling Preemptive Tasks

We consider a simple system consisting of two periodic tasks, Task1 and Task2, executing on a single processor. Task2 has a period of 10 unit of time (u.t.) and a duration of 6 u.t. ; Task1 has a period of 5 u.t. and a duration of 1 and can preempt Task2 at any time. We display in Fig. 4 a TPN model for this system. Our model makes use of a *stopwatch arc*, drawn using an "open box" arrow tip (—□), and of an inhibitor arc (—○).

The net is the composition of four components. The roles of Sched1 and Sched2 is is to provide a token in place psched at the scheduling date of the tasks. The behavior of the nets corresponding to Task1 and Task2 are similar. Both nets are 1-safe and their (unique) token capture the state of the tasks. When the token is in place e, the task execute; when it is in place w it is waiting for its scheduling event. Hence we have a scheduling error if there is a token in place psched and not in place w.

We use an inhibitor arc between the place e1 and the transition Task2Scheduled to model the fact that Task2 cannot use the processor if Task1 is already running. We use a stopwatch arc between e1 and the transition Task2Finished to model the fact that Task1 can preempt Task2 at any moment. A stopwatch (inhibitor) arc "freezes" the firing date of its transition. Therefore the completion of Task2 (the firing date of Task2Finished) is postponed as long as Task1 is running. Using the same approach, we can define a TPN modeling a system with one preemptive task and n "simple"tasks.

We can define an equivalent model using fickle transitions instead of stopwatch. The idea is to add the duration of Task1 to the completion date of Task2 each time Task1 starts executing (that is Task1Scheduled fires). This can be obtained by removing stopwatch arcs and using for Task2Finished the fickle functions $A(\theta) = B(\theta) = \theta + 1$ when Task1Scheduled fires and the identity otherwise. The resulting dynamic TPN is a trans-

# tasks	2	3	5	10	12
# states (fickle/stopwatch)	84 / 83	208 / 205	1 786 / 1 771	539 902 / 539 391	5 447 504 / 5 445 457
time speedup (fickle/stopwatch)	×2.00 (0.005s/0.010s)	×1.90 (0.022s/0.042s)	×2.12 (0.37s/0.784s)	×2.31 (170s/392s)	×1.95 (3077s/6024s)

Table 1: Comparing the use of Fickle Transitions and Stopwatch.

lation and therefore the *SCG* construction is exact. In this new model, we simulate preemption by adding the duration of the interrupting thread to the completion date of the other running thread. The same idea was used by Bodeveix et al. in [16], where they prove the correctness of this approach using the B method. This scheme can be easily extended to an arbitrary set of preemptive tasks with fixed priority.

The following table gives the results obtained when computing the *SCG* for different number of tasks. The models with fickle transitions have slightly more classes than their stopwatch counterpart. Indeed, in the fickle case, the firing date of Task2Finished can reach a value of 7, while it is always bounded by 6 with stopwatches. The last row of the Table gives the computation time speedup between our implementation of fickle transitions and the default implementation of stopwatch in Tina. We observe that the computation with fickle transitions is (consistently) two times faster; this is explained by the fact that the algorithmic for stopwatches is more complex. Memory consumption is almost equal between the two versions approaches, with a slight advantage for the fickle model.

4.2. Verification of Linear Hybrid systems

The semantics of fickle transitions came naturally from our goal of implementing the QSS method using TPN (see Sect. 2.3). We give some experimental results obtained using this approach on two very simple use cases.

Our first example is a model for the behavior of hydraulic cylinders in a landing gear system [8]. The system can switch between two modes, extension and retraction. The only parameter is the position x of the cylinder head. (It is possible to stop and to inverse the motion of a cylinder at any time.) The system is governed by the relation $\dot{x} = 5 - x$ while opening, with $x \in [0, 5]$, and $\dot{x} = -1$ while closing. We can model this system using two fickle transitions.

The second example is a model for a *double integrator*, an extension of the simple integrator of Fig. 1 to a system with two interdependent variables x_1 and x_2. The system has two components, P_1, P_2, where P_i is in charge of the evolution of x_i, for $i \in \{1, 2\}$, and each x_i is governed by the relation $\dot{x_i} = f_i(x_1, x_2)$. The components P_1 and P_2 are concurrent and interact with each other by sending an event when the value of x_i changes. Therefore the system mixes message passing and hybrid evolution. This system can be used to solve second order linear differential equations of the form $\ddot{y} = k_P\dot{y} + k_I(S - y)$; we simply take $\dot{x_1} = x_2$ and $\dot{x_2} = k_P x_2 + k_I(S - x_1)$. This family

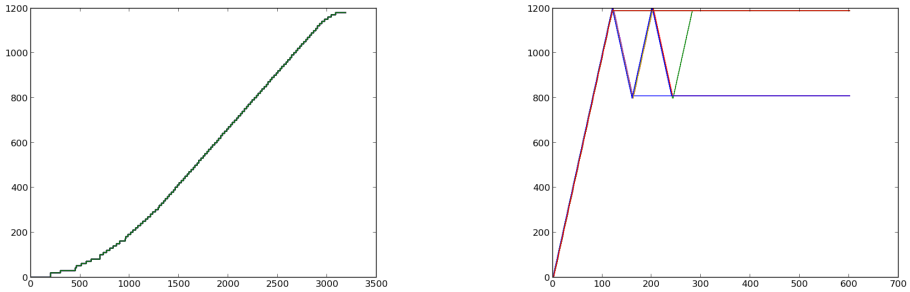


Figure 5: Evolution of the PI-controller: fickle (left) and discrete (right) versions.

of equations often appears in control-loop feedback mechanisms, where they model the behavior of proportional-integral (PI) controller. For example, a system with double quantized integrators is studied in [13] in the context of a dynamic cruise controller.

We compare the results obtained with our two versions of the integrator: fickle and discrete (synchronous). Figure. 5 displays the evolution of the variable x_1 in the PI-controller for our two models, with a quantum of $1/10$. We observe that the discrete version does not converge with this time step (we need to choose a value of $1/100$).

System (version) parameters	Landing Gear (fickle) $Q = 1/10$	Landing Gear (discrete) $h = 1/10$	Cruise Control (PI-controller) (fickle) $Q = 1/10$	Cruise Control (PI-controller) (discrete) $h = 1/10$	Cruise Control (PI-controller) (discrete) $h = 1/100$
# states	1 906	2 590	259	2 049	20 549
time (s)	0.076	0.125	0.004	0.017	0.185
memory (MB)	1.00	1.56	0.11	0.90	9.02

5. Conclusion and Related Work

We have shown how to extend the *SCG* construction to handle fickle transitions. The *SCG* is certainly the most widely used state space abstraction for Time Petri nets: it is a convenient abstraction for LTL model checking; it is finite when the set of markings is bounded; and it preserves both the markings and traces of the net. The results are slightly different with dynamic TPN, even for the restricted class of translation nets. In particular, we may have an infinite *SCG* even when the net is bounded. This may be the case, for instance, if we have a transition that can stay persistent infinitely and that is associated to the fickle function $\mathbf{I}_d(\theta) = [\theta + 1, \theta + 1]$. This entails that our construction may not terminate, even if the set of markings is bounded. This situation is quite comparable to what occurs with *updatable timed automata* [9] and, like in this model, it is possible to prove that the model-checking problem is undecidable in the general case. This does not mean that our construction is useless in practice, as we show in our examples of Sect. 4.

The notion of fickle transitions came naturally as the simplest extension of TPN able

to integrate the Quantized State System (QSS) method [10] inside Tina. Although there are still problems left unanswered, this could provide a solution for supporting hybrid systems inside a real-time model-checker. Theorem 2 gives clues on how to support fickle transitions in existing tools for standard TPN. Indeed, the incremental computation of the coefficients of the "difference-bound matrices" (α, β and γ) is not very different from what is already implemented in tools that can generate a *SCG* for a standard TPN. In particular, the "intermediate" domain D_{int} computed in (C1) is exactly the domain obtained from D in a standard TPN. We only need two added elements. First, we need to apply a numerical function over the coefficients of D_{int}; this is easy if the tool already supports associating a function to a transition in a TPN (as it is the case with Tina). Next, we need to compute the maximal value of a numerical functions over a given interval; this can be easily added to the tool or delegated to a numerical solver. Actually, for the examples presented in Sect. 4, we only need to use affine functions, for which the maximal value can be defined by a straightforward arithmetical expression. As a result, it should be relatively easy to adapt existing tools to support the addition of fickle transitions. This assessment is supported by our experience when extending Tina; once the semantics of fickle transitions was stable, it took less than a week to adapt our tools and to obtain the first results.

To our knowledge, updatable TA is the closest model to dynamic TPN. The relation between these two models is not straightforward. We consider very general update functions but do not allow the use of multiple firing dates in an update (that would be the equivalent of using other clocks in TA). Also, the notion of persistent transitions does not exist in TA while it is central in our approach. While the work on updatable TA is geared toward decidability issues, we rather concentrate on the implementation of our extension and its possible applications. Nonetheless, it would be interesting to define a formal, structural translations between the two models, like it was done in [2, 6] between TA and TPN. Some of our results also show similarities between fickle transitions and the use of stopwatch [4]. In the general case, it does not seem possible to encode one extension with the other, but it would be interesting to look further into this question. Finally, since the notion of slope is central in our implementation of the QSS method (see Sect. 2.3), it would be interesting to compare our results with an approach based on multirate transitions [12], that is a model where time does not advance at the same rate in all the transitions.

For future works, we plan to study an extension of our approach to other models of real-time systems and to other state-space abstractions. For instance the Strong *SCG* construction of [1], that is finer than the *SCG* construction but that is needed when considering the addition of priorities. The strong *SCG* relies on the use of *clock domains*, rather than firing domains, and has some strong resemblance with the zone constructions commonly used for analysis of TA. Another, quite different, type of abstractions rely on the use of a discrete time semantics for TPN. We can obtain a discrete semantic by, for instance, restricting continuous transitions $\xrightarrow{\theta}$ to the case where θ is an integer. This approach could be useful when modeling hybrid systems, since it is a simple way to add a quantization over time as well as over values.

References

[1] B. Berthomieu and F. Vernadat. State class constructions for branching analysis of Time Petri Nets. In *TACAS2003*, volume LNCS2619, page 442. Springer, 2003.

[2] B. Bérard and F. Cassez. Comparison of the expressiveness of timed automata and time petri nets. In *Proc. FORMATS05, vol. 3829 of LNCS*, pages 211–225. Springer, 2005.

[3] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Engineering*, 17(3):259–273, 1991.

[4] B. Berthomieu, D. Lime, O.H. Roux, and F. Vernadat. Reachability problems and abstract state spaces for time Petri nets with stopwatches. *Journal of Discrete Event Dynamic Systems*, 17:133-158, 2007.

[5] B. Berthomieu and M. Menasche. A state enumeration approach for analyzing time Petri nets. In *Proc. Applications and Theory of Petri Nets (ATPN'82)*, pages 27–56, 1982.

[6] B. Berthomieu, F. Peres, and F. Vernadat. Bridging the gap between timed automata and bounded time petri nets. In *Formal Modeling and Analysis of Timed Systems (FORMATS'06), Springer LNCS 4202*, pages 82–97, 2006.

[7] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool TINA – construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14):2741–2756, 15 July 2004.

[8] Frédéric Boniol and Virginie Wiels. The Landing Gear System Case Study. In *ABZ Case Study*, volume 433 of *Communications in Computer Information Science*. Springer, 2014.

[9] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(23):291–345, 2004.

[10] Francois E. Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, 2006.

[11] François E Cellier, Ernesto Kofman, Gustavo Migoni, and Mario Bortolotto. Quantized state system simulation. *Proc. GCMS08, Grand Challenges in Modeling and Simulation*, pages 504–510, 2008.

[12] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with kronos. In *Proc. 1995 IEEE Real-Time Systems Symposium, RTSS'95*, pages 66–75. IEEE Computer Society Press, 1995.

[13] Damien Foures, Vincent Albert, and Alexandre Nketsa. Formal compatibility of experimental frame concept and FD-DEVS model. *Proc. of MOSIM'12, International Conference of Modeling, Optimization and Simulation*, 2012.

[14] Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier H Roux. Romeo: a tool for analyzing time petri nets. In *Proceedings of the 17th international conference on Computer Aided Verification*, pages 418–423. Springer, 2005.

[15] P. M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, Department of Information and Computer Science, University of California, 1974.

[16] Odile Nasr, Miloud Rached, Jean-Paul Bodeveix, and Mamoun Filali. Spécification et vérification d'un ordonnanceur en B via les automates temporisés. *L'Objet*, 14(4), 2008.

[17] G. Ramalingam, J. Song, L. Joscovicz, and R. E. Miller. Solving difference constraints incrementally. *Algorithmica*, 23, 1995.

[18] Enrico Vicario, Luigi Sassoli, and Laura Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. Software Eng.*, 35(5):703–719, 2009.

A. Proof of Theorem 1

Theorem 1: *For every weak DTPN, N, with a finite set of reachable markings, there is a TPN, $N^{\times}$, with an equivalent semantics.*

We say that two nets have equivalent semantics if their state graphs are weakly timed bisimilar (see Def. 4 below).

We assume that N is the weak DTPN $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0, \mathbf{I}_s, \mathbf{I}_d \rangle$. Since N is weak, the function $\mathbf{I}_d$ is trivial and the behavior of persistent transitions is the same than for TPN. By hypothesis, we also have that the set of markings of N, say $\mathcal{M}$, is bounded.

We define a 1-safe TPN $N^{\times}$ that will simulate the execution of N. Some places of $N^{\times}$ will be used to denote the marking in the net N. We denote $P_{\mathcal{M}}$ the set containing one place for every marking in $\mathcal{M}$. We use the same symbol, m, to denote the place and the marking. The places of $P_{\mathcal{M}}$ are a subset of the places of $N^{\times}$.

Since a TPN is also an example of weak DTPN, our construction can be used in order to find a 1-safe TPN equivalent to any given (bounded) TPN.

Corollary 1. *For every TPN, with a finite set of reachable markings, there is a 1-safe TPN with an equivalent semantics.*

The definition of $N^{\times}$ is based on the composition of a collection of TPN, denoted $E(t, I)$, that models the situation where the transition t of N is currently enabled and where the firing date of t was picked in the time interval I. Therefore I belongs to the set of time intervals, denoted $\mathbb{I}_{\mathcal{M}}$, that can appear during the evolution of N. The set $\mathbb{I}_{\mathcal{M}}$ is also finite and has less than $|T|.|\mathcal{M}|$ elements.

$$\mathbb{I}_{\mathcal{M}} = \{\ \mathbf{I}_s(k, m) \mid m \in \mathcal{M}, k \in T\ \}$$

When dealing with a particular transition t, we can restrict to time intervals of the form $\mathbf{I}_s(t, m)$ where t is enabled at m.

$$\mathbb{I}(t) = \{\ \mathbf{I}_s(t, m) \mid t \in \mathcal{E}(m)\ \}$$

Before defining formally $N^{\times}$, we start by defining some useful notations and by giving some intuitions on our encoding.

A.1. Definitions and Useful Notations

We define the TPN $E(t, I)$ for every pair (t, I) of a transition t in T and a time interval I in $\mathbb{I}(t)$. We give a graphical description of the net in Fig. 6.

The net $E(t, I)$ has two places $p_{t,I}$ and $q_{t,I}$. The place $p_{t,I}$ is the *initial place* of $E(t, I)$. Intuitively, we will place a token in $p_{t,I}$ when the transition t becomes newly-enabled by a marking, say m, and $I = \mathbf{I}_s(t, m)$. The token moves to $q_{t,I}$ when the transition has been enabled for long enough, that is when we reach the firing date of t. Hence the purpose of transition t_I is to record the timing constraint associated to t. This transition is "local" to $E(t, I)$, meaning that no other places in $N^{\times}$ has access to it.

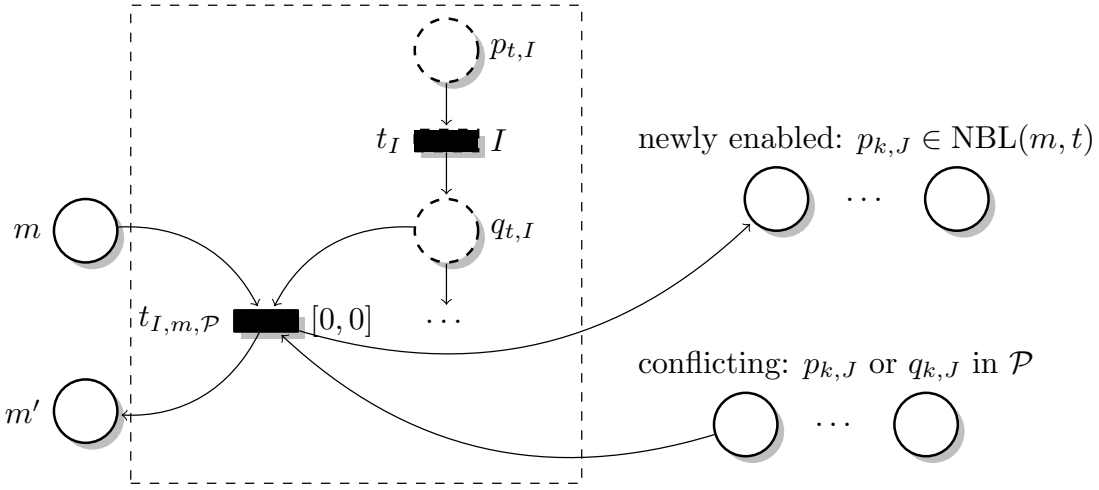


Figure 6: The TPN $E(t, I)$

The final ingredient in the definition of $E(t, I)$ is a collection of transitions $t_{I,m,\mathcal{P}}$, where the marking m enables t, that is t is in $\mathcal{E}(m)$. These transitions have timing constraints $[0, 0]$ and can empty the token in place $q_{t,I}$. The purpose of the transition $t_{I,m,\mathcal{P}}$ is to model the firing of transition t from the marking m in N. In particular, a transition $t_{I,m,\mathcal{P}}$ will empty the place m of $P_{\mathcal{M}}$ and put a token in the place m' such that $m \xrightarrow{t} m'$, that is $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$.

When the transition $t_{I,m,\mathcal{P}}$ fires, it should also "enable" new transitions and "disable" the transitions that are in conflict with t in N. More precisely, the transition should: (1) put a token on the initial place of the net $E(k, J)$, where $k \in \mathrm{nbl}(m, t)$ and $J = \mathbf{I}_s(k, m')$; and (2) remove the token from the net $E(k', J')$ such that k' was enabled at m but not at m' (conflicting transitions). The transitions of N that are persistent when t fires are not involved; therefore their firing date are left untouched. The treatment of conflicting transitions is quite complex. Indeed, it is not possible to know exactly the time interval J' associated to k' and therefore we should test all possible combinations. Another source of complexity is that the token in $E(k', J')$ can be either in the initial place, $p_{k',J'}$, or in the place $q_{k',J'}$ (The parameter $\mathcal{P}$ is used to differentiate the multiple choices.)

We define the predicate $\mathrm{NBL}(m, t)$ that describes the set of initial places of the nets $E(k, J)$ such that k is newly-enabled after t fires from m.

$$\mathrm{NBL}(m,t) = \left\{ p_{k,J} \mid k \in \mathrm{nbl}(m,t) \wedge \left(m \xrightarrow{t} m'\right) \wedge \left(J = \mathbf{I}_s(k,m')\right) \right\}$$

Likewise, we define the predicate $\mathrm{CFL}(t, m)$ that describes sets of places in the net $E(k, J)$ such that k conflicts with t at marking m. The definition of CFL relies on the relation $t \bowtie_m k$, meaning that k and t are in conflict in the marking m, that is $m \not\geq \mathbf{Pre}(t) + \mathbf{Pre}(k)$. A set of places $\mathcal{P}$ is in $\mathrm{CFL}(t, m)$ if it has exactly one place in

each transition in conflict with t.

$$\bowtie(m,t) \;=\; \{k \mid m \not\geq \mathbf{Pre}(t) + \mathbf{Pre}(k)\}$$

$$\mathrm{CFL}(m,t) \;=\; \{\{r_{k_1,J_1},\ldots,r_{k_n,J_n}\} \mid r \in \{p,q\} \wedge \bowtie(m,t) = \{k_1,\ldots,k_n\} \wedge J_1 \in \mathbb{I}(k_1) \wedge \cdots \wedge J_n \in \mathbb{I}(k_n)\}$$

There are at most $2^{|T|}$ sets in $\mathrm{CFL}(m,t)$. Next, we use all these predicates to formally define the net $E(t,I)$ and, ultimately, the TPN $N^{\times}$.

Definition 2. *The net $E(t,I)$ is the 1-safe TPN such that:*

- *the set of places is $P^{\times}$;*

$$P^{\times} = P_{\mathcal{M}} \cup \{p_{t,I}, q_{t,I} \mid t \in T, I \in \mathbb{I}(t)\}$$

- *the set of transitions is $\{t_I\} \cup \{t_{I,m,\mathcal{P}} \mid t \in \mathcal{E}(m), \mathcal{P} \in \mathrm{CFL}(m,t)\}$;*
- *the pre- and postconditions of t_I are as in Fig. 6;*
- *the places in the precondition of $t_{I,m,\mathcal{P}}$ are $\{q_{t,I}, m\} \cup \mathcal{P}$*
- *the places in the postcondition of $t_{I,m,\mathcal{P}}$ are $\{m'\} \cup \mathrm{NBL}(m,t)$*
- *the static time interval of t_I is I and of the transitions $t_{I,m,\mathcal{P}}$ is $[0,0]$;*
- *there is no token in the net in the initial marking;*

All the nets $E(t,I)$ have the same set of places. The net $N^{\times}$ is the 1-safe TPN obtained by the "union" of the nets $E(t,I)$; places with the same identifier are fusioned and transitions are not composed.

Definition 3. *The net $N^{\times}$ is the 1-safe TPN such that:*

- *the set of places is $P^{\times}$, as in Def. 2;*
- *the set of transitions is $T^{\times}$;*

$$T^{\times} = \{t_I, t_{I,m,\mathcal{P}} \mid m \in \mathcal{M}, t \in \mathcal{E}(m), I \in \mathbb{I}(t), \mathcal{P} \in \mathrm{CFL}(m,t)\}$$

- *the static time interval and the pre- and postconditions of the transitions in $T^{\times}$ are as in Def. 2;*
- *in the initial marking there is one token in each place $p_{t,I}$ such that t is enabled at m_0, the initial marking of N, and $I = \boldsymbol{I}_s(t, m_0)$;*

Our encoding of N is not very concise. Indeed, the best bounds for the size of $N^{\times}$ are in $O(|T|.|\mathcal{M}|)$ for the number of places and in $O(2^{|T|}.|T|.|\mathcal{M}|^2)$ for the number of transitions. We can strengthen these bounds if N is a TPN, that is when there is only one possible time interval for each transition. In this case the bound for the number of places is $O(|T| + |\mathcal{M}|)$ and the bound for the number of transitions is in $O(|T|.2^{|T|})$. We can also choose a more concise representation for the markings; such that we use a vector of places to encode the possible markings (in binary format) instead of using one place for every single marking (a representation in unary format).

A.2. Correctness of our Encoding

We start by recalling the notion of (weak) timed similarity between Timed Transition Systems (TTS) (see for example [The Expressive Power of Time Petri Nets, Bérard et al, 2012]). We consider a distinguished set of actions that stands for "silent/unobservable events"; we assume that every silent action as the label τ. The weak transition relation $\stackrel{\alpha}{\Rightarrow}$ is defined as $(\stackrel{\tau}{\rightarrow})^* \stackrel{\alpha}{\rightarrow}$ if $\alpha \neq \tau$ and as $(\stackrel{\tau}{\rightarrow})^*$ otherwise. Hence we always have $s \stackrel{\tau}{\Rightarrow} s$ for every state s.

Definition 4. *Assume $SG_1 = \langle S_1, S_0^1, \rightarrow_1 \rangle$ and $SG_2 = \langle S_2, S_0^2, \rightarrow_2 \rangle$ are two TTS. A binary relation $\mathcal{R}$ over $S_1 \times S_2$ is a* weak timed simulation *if, whenever $s_1 \stackrel{\alpha}{\rightarrow}_1 s_1'$ in SG_1 then for every state $s_2 \in S_2$ such that $s_1 \mathcal{R} s_2$ there is a state s_2' such that $s_2 \stackrel{\alpha}{\Rightarrow}_2 s_2'$ and $s_1' \mathcal{R} s_2'$.*

We say that two TTS are (weakly timed) bisimilar, denoted $SG_1 \approx SG_2$, if there is a binary relation $\mathcal{R}$ over $S_1 \times S_2$ such that both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are weak timed simulations.

Next we show that SG, the state graph of N, and $SG^\times$, the state graph of $N^\times$, are bisimilar. The definition of $\approx$ depends implicitly on the definition of the silent events τ. In our case, the only silent actions correspond to the discrete events t_I in $T^\times$. Intuitively, an action of the form t_I only indicates that the transition $t \in T$ has reached its firing date. It has no effect on the marking (places in $P_\mathcal{M}$) or on the other nets $E(k, J)$. On the opposite, an action $t_{I,m,\mathcal{P}}$ commits the decision to fire t. We use the action t to refer to any transition of the kind $t_{I,m,\mathcal{P}}$ in $SG^\times$.

We list a sequence of properties on the semantics of $N^\times$. Since this a 1-safe net, we say that a place r is marked on a state $(m^\times, \varphi^\times)$ of $N^\times$ if $m^\times(r) = 1$. The following properties hold on every reachable state in $SG^\times$:

1. there is only one token marked in the places $P_\mathcal{M}$;
2. if m is marked then there is only one token among the collection of (sub)nets $E(t, I)$, for every $t \in \mathcal{E}(m)$. Moreover there are no token in the net $E(k, J)$ if $k \notin \mathcal{E}(m)$;
3. if the places m and $q_{t,I}$ of $E(t, I)$ are marked then there is exactly one set $\mathcal{P}$ in $\mathrm{CFL}(m, t)$ such that $t_{I,m,\mathcal{P}}$ is enabled; this is the only transition enabled in $E(t, I)$.

We can prove these properties by induction on the sequence of transitions (the path) from the initial state of $SG^\times$ to a state. If the net $E(t, I)$ is marked, it means that the timing constraints of t, at the time it was newly enabled, was I.

We define an interpretation function $[\![.]\!]$ between states of $SG^\times$ and states of SG. Assume $s_\times = (m^\times, \varphi^\times)$ is a state in $SG^\times$, then $[\![s_\times]\!]$ is the state (m, φ) such that:

- the marking m corresponds to the only place of the kind $m \in P_\mathcal{M}$ that is marked in $m^\times$ (see property 1 above);

- for every $t \in \mathcal{E}(m)$ there is a unique net $E(t, J)$ marked in $N^\times$ (see property 2 above), then if the place $p_{t,J}$ is marked we have $\varphi(t) = \varphi^\times(t_{t,J})$ and if $q_{t,J}$ is marked we have $\varphi(t) = 0$.

We observe that, with our interpretation, the initial states of $SG^\times$ is mapped to the initial state of SG. Again, using an induction on the paths of $SG^\times$, it is possible to prove that every state of the form $[\![s_\times]\!]$ is reachable in N. Conversely, we prove that every state in SG has a counterpart in $SG^\times$. Actually, we prove a stronger property that will be useful to prove the equivalence between state graphs.

Lemma 1. *For every state $s \in SG$ there is a state $s_\times \in SG^\times$ such that $s = [\![s_\times]\!]$ and for every action $\alpha \in T \cup \mathbb{R}_{\geq 0}$; if $s \xrightarrow{\alpha} s'$ then there is a state $s'_\times$ in $SG^\times$ such that $s_\times \overset{\alpha}{\Rightarrow} s'_\times$ and $s' = [\![s'_\times]\!]$.*

sketch. By induction on the sequence of transitions from the initial state s_0 of SG to s. We already observed that s_0 is the interpretation of the initial state of $SG^\times$. Assume that $s = (m, \varphi)$ has a counterpart $s_\times$ in $SG^\times$ and that $s \xrightarrow{\alpha} s'$.

We first study the case of discrete transitions, that is $\alpha = t$ with $t \in \mathcal{E}(m)$. Assume $E(t, I)$ is the net marked in $s_\times$ that corresponds to t. Since there is a discrete transition from s, we have that $\varphi(t) = 0$, which means that either $tI^\times(t_I) = 0$ (the transition can fire in $N^\times$) or that $q_{t,I}$ is already marked. Then there is a unique set $\mathcal{P}$ such that $t_{I,m,\mathcal{P}}$ can fire, and it can fire immediately. This means that there is a state $s'_\times$ such that $s_\times \overset{t}{\Rightarrow} s'_\times$. By definition of $t_{I,m,\mathcal{P}}$, we can choose the same firing dates for the newly enabled transitions in $s'_\times$ than in s', hence $s' = [\![s'_\times]\!]$.

Assume that α is a continuous action $\theta \in \mathbb{R}_{\geq 0}$. We need to prove that we can let the time elapse of θ in the state $SG^\times$. We can assume that $\theta \neq 0$, otherwise $s' = s$. Since we can let θ elapse from s we have that $\theta \leq \varphi(t)$ for every transition $t \in \mathcal{E}(m)$. By definition of $[\![.]\!]$ we have that $\varphi(t) = \varphi^\times(t_I)$ for some interval $I \in \mathbb{I}(t)$. Then we also have $s_\times \overset{\theta}{\Rightarrow} s'_\times$ and $s' = [\![s'_\times]\!]$. □

Our candidate relation $\mathcal{R}$ for the bisimulation is the binary relation from $SG \times SG^\times$ such that $s \mathcal{R} s^\times$ if and only if $s = [\![s^\times]\!]$. From our previous results we already have that $\mathcal{R}$ is total. Hence we just need to prove that both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations. The property for $\mathcal{R}$ is a direct corollary of Lemma 1. For the inverse relation, we assume that $s_\times \xrightarrow{\alpha} s'_\times$ in $SG^\times$. We have three possible case for the action α. The case $\alpha = \tau$ is trivial since, in this case, we have $[\![s_\times]\!] = [\![s'_\times]\!]$. The cases where α is a discrete transition t or a continuous transition $\alpha \in \mathbb{R}_{\geq 0}$ is similar than for Lemma 1. Hence SG and $SG^\times$ are weakly time bisimilar. QED.