INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On equivalence between Timed State Machines and Time Petri Nets

Stefan Haar, Laurent Kaiser, Françoise Simonot-Lion, and Joel Toussaint

N° 4049

November 8, 2000

THÈME 1

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4049--FR+ENG

On equivalence between Timed State Machines and Time Petri Nets

Stefan Haar*, Laurent Kaiser, Françoise Simonot-Lion†, and Joel Toussaint‡

Thème 1 — Réseaux et systèmes
Projet TRIO

Rapport de recherche n° 4049 — November 8, 2000 — 17 pages

Abstract: In this article, we identify a subclass of Timed Automata (Alur and Dill [1]), called *Timed State Machines* as weakly equivalent w.r.t. strongly timed behavior to a canonical class of Time Petri Nets (TPNs) in the sense of Merlin and Farber [17]; more precisely, the weak equivalence holds for bounded non-Zeno TPN with self-concurrency 1, denoted N1TPN. TSMs, and in particular message synchronized products, called TIOSMs, are mainly used for test generation; on the other hand, there is a rich literature on TPNs in verification. Hence our motivation to combine the strengths of both models. We present here an explicit construction for two - way translation between 1-TPNs and TSMs; in both directions, the power of clock timing is exploited to obtain concise and analyzable models. The TSM model obtained from the translation has a state set the size of the reachability graph; it thus improves on the class graph obtained by the *enumerative method* [3], [2]. The existence of the translation procedure, which has also been implemented in a tool prototype, *XTIOSM*, makes the model equivalence effective.

Key-words: Timed Automata, Time Petri Nets, real time systems, test generation, verification

* École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05, France, email: Stefan.Haar@ens.fr.
Supported by the European TMR project ALAPEDES, Contract Reference ERB-FMRX-CT-96-0074.
Corresponding Author

† LORIA-ENSEM, 2 avenue de la Forêt de la Haye, 54516 Vandoeuvre, France.
Email: {Laurent.Kaiser, Francoise.Simonot}@loria.fr

‡ Univ. Blaise Pascal, Clermont-Ferrand, France. Email: toussaint@lirep.univ-bp.clermont.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Sur l'équivalence entre les *Timed State Machines* and les Réseaux de Pétri temporels

Résumé : Dans cet article, nous identifions une sous-classe nommée *Timed State Machines* des automates temporisés de Alur et Dill [1] comme faiblement equivalente (par rapport au comportement fortement temporisé) à une classe canonique de Réseaux de Pétri temporels proposés Merlin et Farber [17]; plus précisement, cette faible équivalence est établie avec les RdP temporels bornés, *non-Zeno* et à auto-concurrence 1, notés N1TPN. Les TSM, et notamment leus produits synchronisés à messages qu'on nomme TIOSM, sont surtout utilisés pour générer des tests. De l'autre part, il existe une littérature très riche sur l'utilisation des RdP temporels en vérification. D'où notre motivation de combiner la puissance des deux modèles.

Nous montrons ici une construction explicite qui permet de traduire les N1TPN en TSM et vice versa; dans les deux sens, on s'appuye sur la puissance de la temporisation par horloges pour obtenir des modèles concis et analysables. Le graphe d'états du TSM obtenu par la traduction et de la taille du graphe d'atteignabilité; il est donc plus efficace que le graphe de classes obtenu par la *méthode énumérative* [3], [2]. L'existence du processus de traduction, implanté au sein d'un prototype d'outil, *XTIOSM*, rend l'équivalence des modèles effective.

Mots-clés : Automates temporisés, Réseaux de Pétri temporels, sytèmes temps réel, génération de tests, vérification

1 Introduction

While there exist, for Petri Nets without timing, powerful partial order techniques using net unfoldings (cf. [8], [16], [7]) for formal verification, similar approaches have only a limited impact in the domain of *Time Petri Nets (TPN)* introduced by Merlin and Farber [17]; cf., however, Lilius [9]). The analysis of such systems therefore generally relies on (some version of) the *access graph* and amounts thus to an analysis of timed automata in the sense of Alur and Dill [1]. This fact, together with the interest in using analysis tools designed for one model in the context of the other, motivated us to investigate the equivalence of both model classes and to look for an automatic translation in both ways. The second and third sections present the two formalisms. In preparing for the translation procedure, we have a closer look at timing constraints and timed semantics: we identify the suitable subclasses of timed automata and time PN, as well as the proper notion of timed behavior to allow the comparison. The notion of observational equivalence is introduced in the forth section; the translation procedure respecting observational equivalence is given in Sections 5 and 6.

2 Timed Automata and Timed State Machines

2.1 Clock Constraints

Let $\mathbb{T} \subseteq \mathbb{R}$ be a *time domain*[1]; set $\mathbb{T}_+ := \mathbb{T} \cap [0, +\infty)$. For a set C of clock variables, c a variable with values in C and x in[2] $\mathbb{T}_+$, define the sets $\Phi(C)$ and $\Psi(C)$ of clock constraints γ as (we omit parentheses):):

$$\begin{array}{rcl} \text{for } \Phi(C): \ \gamma & := & x \leqslant c \mid c \leqslant x \mid \neg\gamma \mid \gamma_1 \wedge \gamma_2; \\ \text{for } \Psi(C): \ \gamma & := & x \leqslant c \mid c \leqslant x \mid \gamma_1 \wedge \gamma_2. \end{array}$$

Φ was defined in [1]. Note that $\Psi(C) \subseteq \Phi(C)$, with proper inclusion: neither open intervals nor disjunctions are contained in $\Psi(C)$. We have the following equivalent representation of $\Psi(C)$ and $\Phi(C)$: Let

$$\begin{array}{rcl} \mathcal{I} & := & \{[a,b],\ [a,\infty):\ a,b \in \mathbb{T}_+,\ a \leqslant b\} \\ \tilde{\mathcal{I}} & := & \{[a,b),\ (a,b],\ (a,b),\ (a,\infty):\ a,b \in \mathbb{T},\ a \leqslant b\} \end{array}$$

Then, with c as above and I_1 taking values in $\tilde{\mathcal{I}}$ and I_2 in $\mathcal{I}$:

$$\begin{array}{rcl} \text{for } \Phi(C): \gamma & = & c \in I_1 \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1 \vee \gamma_2 \\ \text{for } \Psi(C): \gamma & = & c \in I_2 \mid \gamma_1 \wedge \gamma_2. \end{array}$$

[1] Alur and Dill [1] have $\mathbb{T} = \mathbb{Q}$ for timed sequences and $\mathbb{T} = \mathbb{R}$ for clock constraints

[2] In our models, no clock will ever read a negative time, so all negative time values can be discarded

2.2 Definition of TAs and TSMs

With the above, we define, following Alur and Dill's [1] definition of *timed transition tables*:

Definition 1 *A* **Timed Automaton (TA)** *is a tuple* $\mathcal{M} = (S, s_0, C, \mathcal{E})$*, where*

1. $S \neq \emptyset$ *is a finite set of* states*,*
2. $s_0 \in S$ *the* initial state*,*
3. $C \neq \emptyset$ *a finite set of* clocks*, all initially set at* 0*,*
4. $\mathcal{E}$ *a set of* actions *having the form* $e = (\mathfrak{st}(e), \mathcal{D}(e), \mathcal{Z}(e), \mathfrak{tg}(e))$*, where:*
   (a) $\mathfrak{st}(e) \in S$ *is the* source state *of* e*,*
   (b) $\mathcal{D}(e)$ *a finite subset of* $\Phi(C)$*,*
   (c) $\mathcal{Z}(e) \subseteq C$ *the set of clocks to be* reset at 0 *at the end of* e*, and*
   (d) $\mathfrak{tg}(e)$ *the* target state *of* e*.*

An action e may be executed in state $\mathfrak{st}(e)$ at time t iff, at time t, *all* constraints in $\mathcal{D}(e)$ are satisfied. Note that the clocks need not show the global time (resets !). Besides that, any clock c may – say, at time t– pass the right hand limit b of, e.g., some $c[a,b]$ in $\mathcal{D}(e)$ without *forcing* the execution of e at t; at t, e may just as well simply loose enabling without any action (*weak timing*). This contrasts with the firing obligation – *strong timing* – in TPNs. We will introduce a more restrictive notion of run for timed automata, called *strong run*, which discards sequences blocked by idling and permits the comparison with TPNs. Also, our construction below leaves the choice to model even the weakly timed TSM behavior by strongly timed 1-TPNs.

Definition 2 (Timed State Machine) *Let* $\mathcal{M}$ *be a timed automaton. If, for every* $e \in \mathcal{E}$*,* $\mathcal{D}(e) = \{\phi_e\}$ *with* $\phi_e \in \Psi$*, then* $\mathcal{M}$ *is called a* **Timed State Machine (TSM)**.

Note that there exist TAs that are not a TSM and cannot be modeled by finite TSMs; Figure 5 shows an example.

2.3 Semantics of Timed Systems

Let Σ be an alphabet with special symbol $\lambda \in \Sigma$; we will use λ as a label for transitions without state changes. We denote the set of non-empty finite words over an alphabet as Σ^+, and $\Sigma^* := \Sigma^+ \cup \{0\}$, where 0 stands for the empty word; the set of *infinite* words is Σ^∞, and $\Sigma^\omega := \Sigma^\infty \cup \Sigma^*$. Let $\theta : \Sigma^\omega \to (\Sigma - \{\lambda\})^\omega$ be the extension of the λ-deleting homomorphism on Σ^*, i.e. the unique mapping such that for all $w_1 \in \Sigma^*$, $w_2 \in \Sigma^\omega$ and $\alpha \in \Sigma - \{\lambda\}$,

$$\theta\left(w_1 \lambda w_2\right) = \theta\left(w_1\right)\theta\left(w_2\right) \quad \text{and} \quad \theta\left(w_1 \alpha w_2\right) = \theta\left(w_1\right)\alpha\theta\left(w_2\right).$$

For a word w, denote its length as $|w|$. For $\sigma \in \Sigma^\omega$, write $\sigma(k\downarrow)$ for the prefix $\sigma_1 \dots \sigma_k$ of σ and $\sigma(k\uparrow)$ for the postfix given by $\sigma = \sigma(k\downarrow)\sigma(k\uparrow)$.

Definition 3 (equivalent to Alur and Dill [1]) *If $\overline{\tau} \in \mathbb{T}^\omega$ satisfies*

1. **Monotonicity**: *For all $i \geq 1$, $\tau_{i+1} \geq \tau_i$, and*
2. **Progress**[3]: *If $|\overline{\tau}| = \infty$, then $\tau_n \stackrel{n\to\infty}{\longrightarrow} \infty$,*

$\overline{\tau}$ is a **timed sequence**. *A* **timed word** *over an alphabet Σ is a pair $\mathfrak{t} = (\sigma, \overline{\tau})$ with $\sigma \in \Sigma^\omega$ and $\overline{\tau}$ a time sequence. Sets of timed words over Σ are called* **timed languages** *over Σ.*

The execution sequences for TA are two-dimensional since the system may advance in the logical domain (by actions) or the time domain (by time steps).

Definition 4 *Let $\mathcal{M} = (S, s_0, C, \mathcal{E})$ be a TA. A* **timed state** *of $\mathcal{M}$ is a pair $\xi = (s, \nu)$ such that $s \in S$ is a state and $\nu \in \mathbb{T}_+{}^{|C|}$ is a vector of clock positions.*

Thus strong enabling does not allow a transition to become disabled without the system changing its logical state. This is a stronger requirement than Alur and Dill's consecution for timed sequences, see below. Since time PNs have strong timing, we can only hope for equivalence if we discard non-strongly timed behavior; besides that, we believe that there is a large class of applications where idling as above can be considered as faulty or incorrectly modeled behavior.

Definition 5 *If $\mathfrak{t} = (\sigma, \overline{\tau})$ is a timed word with $\sigma \in \Sigma^\omega$, a* **run** *of $\mathcal{M}$ over $\mathfrak{t}$ is a sequence $r = r_1 r_2 \ldots$ with $r_i = (\overline{s}_i, \overline{\nu}_i)$,*

$$r: \quad (s_0, \nu_0) \stackrel{\sigma_1, \tau(\sigma_1)}{\longrightarrow} (s_1, \nu_1) \stackrel{\sigma_2, \tau_2}{\longrightarrow} (s_2, \nu_2) \stackrel{\sigma_3, \tau_3}{\longrightarrow} \ldots,$$

where $\overline{s} = s_0 s_1 \ldots$ is a sequence of states, and $\overline{\nu} = \nu_0 \nu_1 \ldots$ a sequence of clock vectors satisfying:

1. **Initialization**: *$s_0 \in S_0$, and $\nu_0(x) = 0$ for all $x \in C$;*
2. **Consecution**: *for all $i \geq 1$, $\mathcal{E}$ contains some $e = (\mathfrak{st}(e), m, \mathcal{D}(e), \mathcal{Z}(e), \mathfrak{tg}(e))$, where $(\nu_{i-1}(x) + \tau_i - \tau_{i-1})_{x \in C}$ satisfies $\mathcal{D}(e)$, and*

$$\nu_i(x) \quad = \quad \begin{cases} 0 & : \quad x \in \mathcal{Z}(e) \\ \nu_{i-1}(x) + \tau(a_i) - \tau(a_{i-1}) & : \quad x \notin \mathcal{Z}(e) \end{cases}$$

Denote the timed language of $\mathcal{M}$ as $\mathcal{L}(\mathcal{M})$ and the set of runs for $\mathcal{M}$ as $\mathfrak{runs}(\mathcal{M})$.

These definitions effectively exclude *Zeno* behavior, i.e. such that no progress is made in spite of an infinite number of time steps with positive duration. But with Alur and Dill's definition, the TA is still allowed to stay in some state s and to let time pass beyond the deadlines of transitions leading out of s; so the system would "abuse" of its weak timing to behave differently from a strongly timed one without any difference in the "reasonable" behavior. To allow for meaningful comparisons, we introduce:

[3] The definition of Alur and Dill [1], "For all $u \in \mathbb{T}$ there exists some $i \geq 1$ such that $\tau(a_i) \geq u$", covers only the case of infinite sequences; the definition is modified to apply to both finite and infinite words.

Definition 6 *Let r with $r_i = (s_i, \nu_i)$ be a run over $\mathfrak{t} = (\sigma, \tau)$ of $\mathcal{M} = (S, s_0, C, \mathcal{E})$. The* **skeleton** *$\rho(r)$ of r is the stuttering free version of s. r is a* **strong run** *if either of the following holds:*

1. $|\theta(\sigma)| = \infty$ *($\Leftrightarrow$ $(\rho(r)| = \infty$), or*
2. $|\theta(\sigma)| =: n \in \mathbb{N}$, *and with $r_k = (s_k, \tau_k)$ the state reached by the last non-λ transition of $\mathfrak{t}$, r_k satisfies: for any e such that $\mathfrak{st}(e) = s_k$ and any $\phi \in \mathcal{D}(e)$,*

$$\left(\bigcap\left[\mathtt{int}(\phi_i) - \nu\left(\mathfrak{clock}\left(\phi_i\right)\right)\right]\right) \cap [0, \infty) \quad = \quad \emptyset, \tag{1}$$

where the ϕ_i are the elementary clauses whose conjunction is ϕ.

The set of strong runs is denoted $\mathfrak{sruns}$, and the language of action sequences corresponding to strong runs is $\mathcal{SL}$.

Condition (1) states that at the time instant at which r reaches for the last time its final state s_k (s_k may have been visited before), all clock constraints for any action leading out of s_k have already expired. Since no clock is reset while the system sits in a state, no future instant will enable any action, so $\mathcal{M}$ is forced to stay in s_k. It is clear that without requiring (1), runs are permitted to simply let all deadlines pass and idle unnecessarily[4]. So equivalence w.r.t. strong runs is finer that run equivalence, and appropriate for comparisons with strongly timed systems; the same holds for weak equivalence, i.e. up to deleting or inserting λ steps.

2.4 TIOSMs: Message synchronization and Tests

A TIOSM is an extension of TSM that permits message synchronization with other TIOSM: (compare *timed cooperating automata* [15]).

Let $\mathcal{M}_i = (S_i, s_0^i, C_i, \mathcal{E}_i)$, $i = 1, \ldots, n$, be TSMs such that $i \neq j$ implies $S_i \cap S_j = \emptyset$, $C_i \cap C_j = \emptyset$, and $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$. Let $\mathfrak{L}$ be a finite set of **messages**, and $m_i : \mathcal{E} \to \{\mu\} \cup (\{!, ?\} \times \mathfrak{L})$ mappings for $i = 1, \ldots, n$. Here, $m_i(e) = \mu$ means that e executes an internal action without communication; $!a$ stands for the *sending* message $a \in \mathfrak{L}$ when e occurs, and $?b$ for *receiving* message $b \in \mathfrak{L}$. The action rules are then complemented by the condition that an action $e \in \mathcal{E}_i$ with $m_i =?a$ can occur only if

1. $\mathcal{M}_i$ is in state $\mathfrak{st}(e)$,
2. $\phi(e)$ is satisfied, and
3. simultaneously, some $e_j \in \mathcal{E}_j$ with $i \neq j$ and $m_j =!a$ occurs,

[4] the pleonasm is intended.

plus the inverse restriction for $m_j = !a$; in particular, actions synchronizing by messages are controlled by more temporal constraints than internal actions since all constraints of their communication partner action have to be fulfilled as well. This synchronization policy reflects *point to point*-synchronization; other policies translate into different rules and, consequently, into a different simulation by Petri Nets as below.

Our interest in TIOSMs here is motivated by their application to testing of real-time applications, used in Kaiser et al. [13], [12] and Koné and Castanet [14]. There are two major branches of applications:

- First, a given specification *Spec* is modeled as a TIOSM; one associates a *test purpose*, i.e. a second TIOSM Tp that represents the property ϕ to be tested, together with a non-empty set $Accept(Tp) \subseteq S(Tp)$ of accepting states. In the synchronous product TIOSM of *Spec* and Tp, one then attempts to calculate on-the-fly a success path π (i.e. leading into $Accept(Tp)$) verifying ϕ. If no such π is found, the property is disproved; a success path π, viewed as a TIOSM itself, yields a *tester* for ϕ. The testers thus obtained are then to be implemented in the application, e.g., an elevator control.
- Multi-component systems can be modeled by TIOSMs components which are then synchronized by means of message exchange; compare the cooperation of timed Automata in [15]. Together with the construction of weakly equivalent TPN described below, this yields a structured modular Time Petri Net modeling technique.

The mutual synchronization of two machines in testing motivates our choice of message synchronization technique.

3 Time Petri Nets, 1-TPNs and N1TPNs

Our interest in TPNs is motivated by the strong results achieved using this formalism in verification, in particular the *enumerative method* of Berthomieu and Diaz [2] and the analysis algorithm presented in [18].

Definition 7 *A* **Petri Net (PN)** $\mathcal{N}$ *is a tuple* $(P, \mathcal{Q}, Pre, Post, M_0)$*, where* $P = P(\mathcal{N})$ *is a finite set of* **places**, $\mathcal{Q} = \mathcal{Q}(\mathcal{N})$ *is a finite set of* **transitions**, *and* $Pre : P \times \mathcal{Q} \to \mathbb{N}$ *and* $Post : \mathcal{Q} \times P \to \mathbb{N}$ *are* **input/output arc functions**, *respectively; a* **marking** *of* $\mathcal{N}$ *is a function* $M : P \times \mathcal{Q} \to \mathbb{N}$*, and* M_0 *is a distinguished marking called the* **initial marking** *of* $\mathcal{N}$*. A transition* q *is* **marking-enabled** *in* M*, denoted* $M \xrightarrow{q}$*, iff* $Pre(p, q) \le M$ *for all* $p \in P$*; the* **firing** *of* q *then transforms* M *into* M'*, denoted* $M \xrightarrow{q} M'$*, where*

$$M' \quad = \quad [M - \Sigma_{p \in {}^\bullet q} Pre(p,q)] + \Sigma_{p \in q^\bullet} Post(q,p). \tag{2}$$

Set $\alpha(M) := \{q \in \mathcal{Q} : \ M \xrightarrow{q}\}$.

In Merlin and Farber [17]'s model, Petri Net transitions are associated with a time interval that describes the period in which an enabled transition may fire. A bullet $^\bullet$ before or after an element denotes its in- or output domain, respectively:

$$\forall p \in \mathcal{P}: \quad {}^{\bullet}p := \{q \in \mathcal{Q}:\ Post(q,p) > 0\},\ p^{\bullet} := \{q \in \mathcal{Q}:\ Pre(p,q) > 0\}$$
$$\forall q \in \mathcal{Q}: \quad {}^{\bullet}q := \{p \in \mathcal{P}:\ Pre(p,q) > 0\},\ q^{\bullet} := \{p \in \mathcal{P}:\ Post(q,p) > 0\}$$

Definition 8 *A* **Time Petri Net (TPN)** *consists of a PN $\mathcal{N}$ and mappings $eft : \mathcal{Q} \to \mathbb{T}_+$ and $lft : \mathcal{Q} \to (\mathbb{T}_+ \cup \{\infty\})$ that assign to each transition its* **earliest** *and* **latest firing times***; we require $eft(q) \le lft(q)$ for all $q \in \mathcal{Q}$. A* **state** *of $\mathcal{N}$ is a pair $s = (M,t)$ with M a marking and $t \in \mathbb{T}_+$ a time instant. Let q become marking-enabled at time τ_0. Then q is* **enabled** *in state $s = (M,\tau)$, $\tau \geqslant \tau_0$, iff*

1. $(\tau - \tau_0) \in [eft(q), lft(q)]$ (**time enabled**), *and*
2. *q was never marking-disabled between τ_0 and τ.*

Write $M \xrightarrow{q,\tau}$ iff q is enabled in $s = (M,\tau)$,and $M \xrightarrow{q,\tau} M'$ iff $M \xrightarrow{q,\tau}$ and $M \xrightarrow{q} M'$

The following will allow semantic comparison of TPN and TA..

Definition 9 *Let $\sigma = q_1 q_2 \ldots \in \mathcal{Q}^{\omega}$. σ is a* **firing sequence** *of $\mathcal{N}$ iff there exist markings M_n, $n \in \mathbb{N}$, of $\mathcal{N}$ such that $M \xrightarrow{q_1} M_1 \xrightarrow{q_2} M_2 \ldots$. If $\overline{\tau}$ is a timed sequence that satisfies $M \xrightarrow{(q_1,\tau_1)} M_1 \xrightarrow{q_2,\tau_2} M_2 \ldots$, we call $(\sigma,\overline{\tau})$ a* **timed firing sequence**. *of $\mathcal{N}$. The set of timed firing sequences for $\mathcal{N}$ is $\mathcal{L}(\mathcal{N})$.*

Associating a clock $c(q)$ to each transition q, we can define timed states, runs and strong runs as above also for TPN; some subtleties with this will be discussed below.

As stated in the introduction, we will focus on *1-TPN*, that is, TPNs with *self-concurrency degree* 1. By this we mean that any given transition q in any reachable state cannot start more than one firing at a time, and, with one firing under way, can not start another before finishing the first. In principle, the equivalences shown below carry over to the bounded case; the problem is that multiple firings of transitions give rise to a variety of distinct firing semantics, for which the adaptation of the proofs necessitate cumbersome and lengthy case distinctions; see [5] for a discussion of multiple firing semantics in TPN. If the case of multiple enabling never occurs, we call $\mathcal{N}$ a *1-TPN* since its maximal degree of self-concurrency is 1. In particular, 1-bounded TPNs are 1-TPNs, but this dynamical restriction is not necessary; self-concurrency degree 1 can be structurally enforced by adding a loop with one place p_q at each q and one token on each p_q in the initial marking. In fact, consider Figure 1: at some time between $\tau \in [a,b]$, q fires; since this firing removes the token from p for one *logical* instant – q's firing not consuming any time, it puts a new token onto p at the same *physical* instant –, the clock for q's next firing is reset at 0, and the next firing will be between $\tau + a$ and $\tau + b$, and so forth.

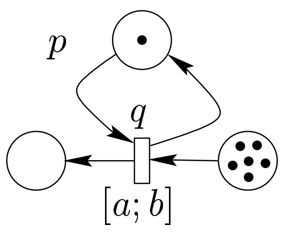

Figure 1: Enforcing self-concurrency degree 1

Moreover, we will have to require *non-Zeno-Ness* for TPNs as we did for TA; call a non-Zeno 1-TPN an **N1TPN**.

4 Observational equivalence of TSM and N1TPNs

In this section, we introduce a notion of observational equivalence for timed systems; this prepares the explicit translations

- for a given timed state machine, into an observationally equivalent 1-TPN with empty transitions, and
- for a given N1TPN $\mathcal{N}$, into an observationally equivalent TSM which is of the size of $\mathcal{N}$'s access graph,

which will be presented in the next two sections.

Let a net or TA be given, and let $\mathfrak{t}^1 = (\sigma^1, \overline{\tau}^1), \mathfrak{t}^2 = (\sigma^2, \overline{\tau}^2) \in \mathbb{T}^\omega$ be two timed words and $r^1 = (s^1, \nu^1)$, $r^2 = (s^2, \nu^2)$ runs over $\mathfrak{t}$ and $\mathfrak{t}^2$, respectively. We say that r^1 is **coarser** than r^2, denoted $r^1 \preceq r^2$, iff (i) $\rho(r^1) = \rho(r^2)$, and (ii) $\tau^1_n \leqslant \tau^2_n$ for all $n \in \mathbb{N}$.

So a coarser run "lumps together" some consecutive time steps that do not change the logical state of the system. Note that this definition allows r^1 to be strictly coarser than r^2 in infinitely many segments of $\mathbb{N}$ simultaneously, so that the two sequences are "infinitely far apart". By contrast, non-Zeno-Ness ensures that there is no infinite stretch of integers $[k, \dots)$ for which $\tau^2_k = \tau^2_{k+n}$ for all n. Therefore,

$$\equiv_\lambda \quad := \quad (\preceq \cup \preceq^{-1})^* \tag{3}$$

defines an equivalence relation on the sets $\mathfrak{runs}$ and $\mathfrak{sruns}$, which we call **observational equivalence**.

5 From TSM to TPNs

This is the more complicated of the two translations between T(IO)SMs and TPNs. We have to pay special attention to the use and reset of clocks, in particular the interaction of several clock constraints of a TSM, and possible communications between two TIOSM components.

5.1 The Backbone Net $\mathcal{N}_\mathcal{M}$

Let $\mathcal{M} = (S, s_0, C, \mathcal{E})$ be a TSM. In the first round of the construction, we ignore all clock constraints; these will be added later on.

1. For every state $s \in S$, create a place $p(s)$; $P_S := \{p(s) : s \in S\}$.
2. For every action $e \in \mathcal{E}$ from state s to s', create a net transition $q = q(e)$. Set $\mathcal{Q}_\mathcal{E} := \{q(e) : e \in \mathcal{E}\}$ and $\mathfrak{l}(q(e)) := e$ (all transitions below will be labeled λ), $eft(q) = 0$ and $lft(q) = \infty$; the clock constraints from $\mathcal{B}$ will be modeled by adding subnets, see below. Further, we add arcs from $p(s)$ to q and from q to $p(s')$, with $Pre(p(s), q) = Post(q, p(s')) = 1$. So the skeleton of our net is an S-net $\mathcal{N}_\mathcal{M} = (P_S, \mathcal{Q}_\mathcal{E}, F_\mathcal{M})$.

3. $M_0(p)$ is set to 1 for $p = p(s_0)$ and to 0 otherwise.

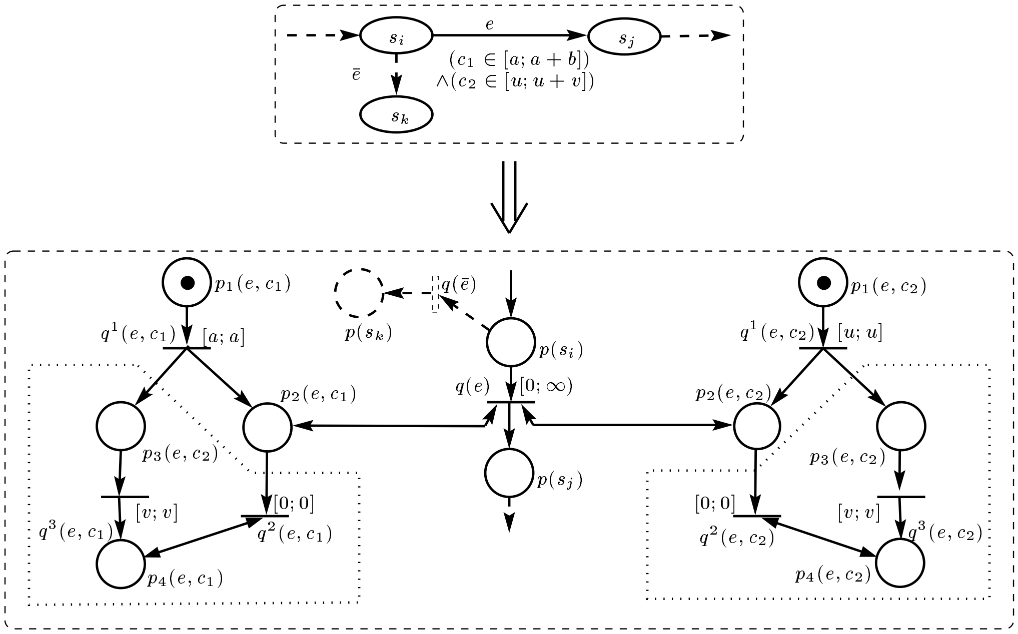

Figure 2: Clock constraints implemented by net modules.

5.2 Translating Clock constraints

We now have to include into the TPN model clock constraints of the form $\bigwedge \phi$, where each ϕ is an elementary clause of the form $c \in [a, b]$ or $c \in [a, \infty)$. For each such elementary ϕ, a subnet $\mathcal{N}_\phi$ representing and ensuring ϕ will be included into $\mathcal{N}$. These *modules*, displayed in Figure 2, consist of four places and three transitions each; for each such module transition q, $\rho(q) = \lambda$. If $lft = \infty$, the modules simplify: In Figure 2, the parts in dashed boxes in each module work exclusively to ensure a finite latest firing time $a+b$; so, for a right-infinite interval (left-infinite once can be ignored since system runs start at time 0) $[a, \infty)$, it suffices to chop away the dashed box of the corresponding module. Note that the modules will have to be complemented to account for clock resets, see below.

Assume first that $I = [a, b]$, i.e. $b = lft(q) < \infty$. Place p_1 is initially marked (since h is at 0); p_2 and p_3 both receive a token at the physical instant when the constraint is first satisfied; so q_i can fire as soon as a token arrives at $p(s_i)$. Then, at time b, p_4 is marked, and p_2 immediately loses its token; q_i is now blocked with respect to the clock constraint $c \in [a, b]$. If $lft(q) = \infty$, remove the transitions q_j^2 and q_j^3 as well as the places p_3 and p_4

(that is, the parts in small dashed boxes in Figure 2); the reasoning is analogous to the above.

Note that the modules in this construction are linked in such a way that no transition from $\mathcal{Q}_{\mathcal{E}}$ is forced to fire while ϕ_e is satisfied; thus the weak timing of TSM is modeled within a strong timing model, that of TPN. Since it can be desirable for test purposes to enforce strong timing, it should be noted that adding priorities is sufficient for this: after slight modifications of the module interplay, give highest priority in any structural conflict to the transition(s) from $\mathcal{Q}_{\mathcal{E}}$; then firing real transitions is prioritized over idling beyond the latest deadline.

5.3 Clock resets

The resetting of clocks makes it necessary to change the preliminary net structure. Note that, in a TIOSM, clocks are reset *after* the transition q_i that requests the reset, but *before* reaching the target state of q. We therefore introduce, for a reset of clock c, a place $p_{reset}(c)$ after $q(q_i)$, along with a new transition $q_{reset}(q_i)$ between $p(s_j)$ and $p_{req-reset}(q_i, c)$.

Further, we complement the clock constraint modules from above as follows (see Figure 3) :

- one place $p^m_{start-reset}(q_i, c)$ for each module m that depends on clock c; we set $Post(q_i, p^m_{start-reset}(c)) = 1$, no other arcs are needed;
- for each clock c concerned, one place $p_{end-reset}(q_i, c)$ with $Pre(p_{end-reset}(q_i, c), q(q_i) = n_c$, where n_c is the number of modules created for clock c.
- in each module m that depends on clock c, three transitions $q(q_i, ca_m)$, $q(q_i, cb_m)$ and $q(q_i, cc_m)$ to implement the reset in the three different possible cases: request received when constraint is
  - *not yet satisfied* (p_1 marked) $\Rightarrow ca_m$,
  - *satisfied* (p_2 and p_3 marked) $\Rightarrow cb_m$, or
  - *no longer satisfied* (p_4 marked) $\Rightarrow cc_m$.

  If $lft(q) = \infty$, transition t_{cc_m} is not needed, and t_{cb_m} has only one ingoing arc since there is no p_3; delete the parts in dashed boxes in Figure 2.

Note that no delay will be caused by the reset module; moreover, $\mathcal{N}$ is 1-safe and therefore a 1-TPN by construction.

Figure 3 shows two clock constraints with one reset. Note that there are $3 \cdot n(c)$ arcs leaving each place $p_{start-reset}(q, c)$ and $3 \cdot n(c)$ arcs entering each place $p_{start-reset}(q, c)$: clock c has to be reset wherever it is present, i.e. in each module representing a c-constraint, not necessarily concerning the same action.

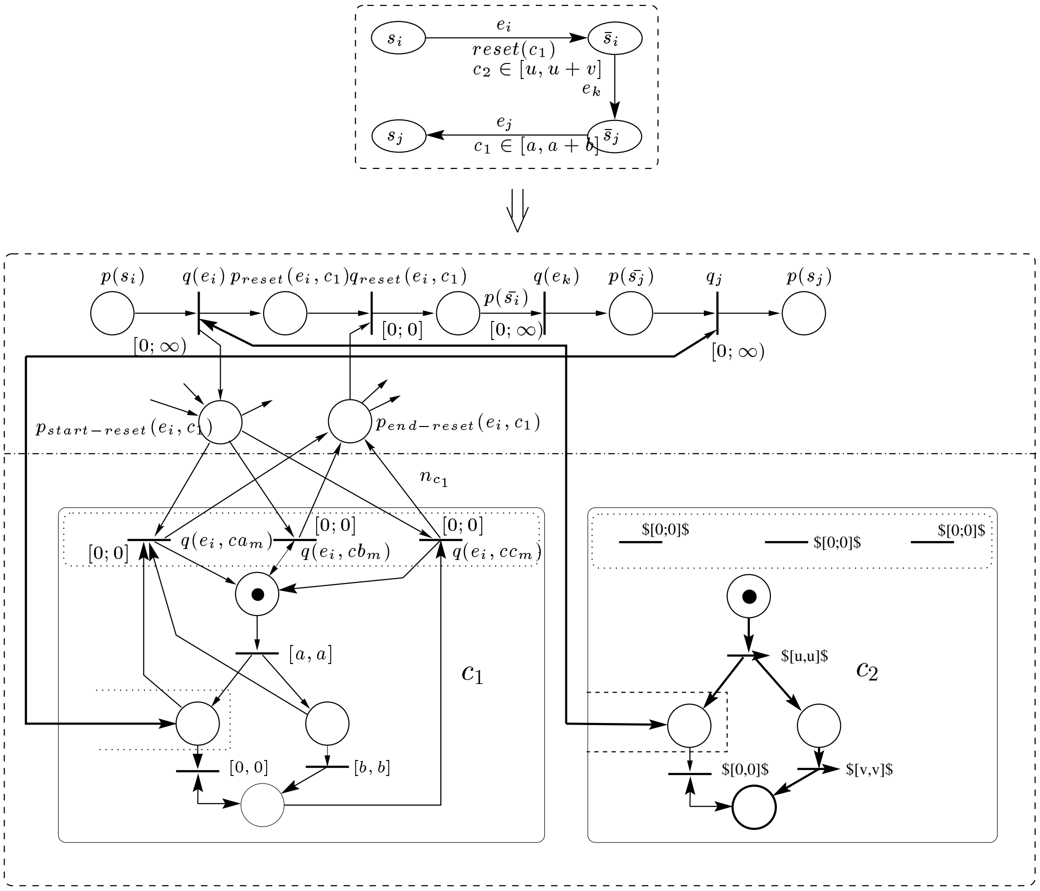

Figure 3: Combining clock resets and constraint modules.

5.4 Equivalence of the Net Model

The comparison of $\mathcal{M}$ and $\mathcal{N}(\mathcal{M})$ has to take into account that only the transitions in $\mathcal{Q}_{\mathcal{E}}$ correspond to actions in $\mathcal{E}$; all others, belonging to clock constraint modules, are labeled λ and must not be regarded as changing the logical state of the net. So, we lump net markings together into equivalence classes: For markings M_1, M_2 of $\mathcal{N}$ set

$$M_1 \equiv_{\mathcal{M}} M_2 \quad :\Longleftrightarrow \quad M_1(p) = M_2(p) \;\forall\; p \in P_S.$$

Then $\equiv_{\mathcal{M}}$ is an equivalence relation. Moreover:

Lemma 1 *For any $\mathfrak{w} = (\sigma_i, \tau_i) \in \mathcal{L}(\mathcal{N})$ and run $\mathfrak{r} \in \mathfrak{runs}(\mathcal{N})$ over $\mathfrak{w}$:*

1. *If* $(M_i, \nu_i) \xrightarrow{\mathfrak{w}_i} (M_{i+1}, \nu_{i+1})$*, then the following holds:*

$$M_i \equiv_{\mathcal{M}} M_{i+1} \quad \Longleftrightarrow \quad q_i \notin P_S,$$

and both imply that $\nu_{i+1} = \nu_i$*. In this case, with* $\bar{\mathfrak{w}} := \mathfrak{w}(i\downarrow)\mathfrak{w}((i+1)\uparrow)$*; then* $\bar{\mathfrak{w}} \equiv \lambda\mathfrak{w}$

2. *For any* $\mathfrak{w} \in \mathcal{L}(\mathcal{M})$ *and a run* $\mathfrak{r} \in \mathfrak{runs}(\mathcal{M})$*, there exist* $\mathfrak{w} \in \mathcal{L}(\mathcal{N})$ *and* $\overline{\mathfrak{w}} \in \mathcal{L}(\mathcal{N}_{\mathcal{M}})$ *such that* $\mathfrak{w} \equiv_{\mathcal{M}} \overline{\mathfrak{w}}$ *and* $\bar{\mathfrak{r}}_i = (M(i), \nu_i)$*, where* $M_i(p) = \delta_{p,p(s_i)}$*.*

Note that $\equiv_{\mathcal{M}}$-equivalent markings correspond to the same state in $\mathcal{M}$.

Now, let $\mathcal{M}$ and $\mathcal{N}(\mathcal{M})$ be in state $\xi = (s, \nu)$.

Theorem 1 *For every* $r^1 = (\sigma^1, \tau^1) \in \mathfrak{sruns}(\mathcal{M})$ *and* $r^2 = (\sigma^2, \tau^2) \in \mathfrak{sruns}(\mathcal{N})$ *there exist* $\bar{r}^1 = (\bar{\sigma}^1, \bar{\tau}^1) \in \mathfrak{sruns}(\mathcal{N})$ *and* $\bar{r}^2 = (\bar{\sigma}^2, \bar{\tau}^2) \in \mathfrak{sruns}(\mathcal{M})$ *such that*

$$r^1 \equiv_\lambda \bar{r}^1 \quad \textit{and} \quad \bar{r}^2 \equiv_\lambda r^2.$$

Proof: $r^1 \to \bar{r}^1$: Suppose $\mathcal{M}$ is in state $\xi = (s, \nu)$ and $\mathcal{N}(\mathcal{M})$ in a state (M, ν) such that $M \equiv_\lambda \delta_{p,p(s)}$. If $\xi \xrightarrow{e,\tau}$ and $\xi' = (\mathfrak{tg}(e))$, write $\xi \xrightarrow{e,\tau} \xi'$. Set

$$poss(\xi) \;\; := \;\; \{e \in \mathcal{E} : \; \mathfrak{st}(e) = s, \textit{and } \nu \textit{ satisfies } \bigwedge_{\phi \in \mathcal{D}(e)} \phi\},$$

and for $t \in [0, \tau]$, set $\nu_t := \nu + (t, \ldots, t)$ and $\xi_t = (s, \nu_t)$. The following facts are immediate for the initial state an extend by induction over runs:

1. If for all $t \in [0, \tau]$, $poss(\xi) = poss(\xi_t)$, then $\mathcal{N}$ can reach a state (M', ν_t) such that $M' \equiv_\lambda M$ and firing only λ-labeled transitions, i.e. a sequence $\sigma \in \mathcal{Q}_\lambda := \mathcal{Q} - \mathcal{Q}^*_{\mathcal{E}}$;
2. if $\xi \xrightarrow{e,\tau} \xi'$ with $\xi' = (s', \nu')$, then $\mathcal{N}$ can reach a state (M', ν') such that $M' \equiv_\lambda \delta_{p,p(s')}$ and firing some $\sigma = \sigma^a q \sigma^b$ with $\sigma^1, \sigma^2 \in \mathcal{Q}^*_\lambda$ and q a transition labeled with e.

In fact, 1 follows by inspection of the clock modules, noting that no reset can occur during the time interval considered as long as no e fires. For 2, note first that $q(e)$ is enabled iff e is. The resets triggered by e transforms ν_t into ν'; the reset device in $\mathcal{N}(\mathcal{M})$ performs the resets in zero physical time, whatever the current state of the clock modules.

Conversely, one obtains a valid run $\bar{r}^2$ from r^2 by grouping together equivalent markings and λ transitions as above in the inverse direction; again, the skeletons of both runs consist solely of $\mathcal{E}$ and $\mathcal{Q}_{\mathcal{E}}$ transitions, respectively. □

As a consequence, $\mathcal{N}$ inherits also non-Zeno-Ness from $\mathcal{M}$; so $\mathcal{N}(\mathcal{M})$ is a N1TPN.

5.5 Modeling Communication: Extension to TIOSMs

In order to extend the above construction to TIOSMs, we have to account for messages. Suppose some message m is used by two components $\mathcal{B}_1$ and $\mathcal{B}_2$, such that $q_1 \in \mathcal{E}(\mathcal{B}_1)$ awaits that message (denoted $?m$). Then, q_1 can fire only jointly with a corresponding transition q_2 from $\mathcal{B}_2$ that sends m (denoted $!m$), and vice versa. Therefore, such pairs of transitions will be represented by a single net transition, the fusion of the component transitions, modeling the synchronization by m. For uniqueness, we thus need to ensure that at most one action per component emits $!m$ if $m(q) = ?m$ for any q in any component. This fusion construction corresponds to *point-to-point-communication*; for other types of communication, the net translation has to, and can, be modified; we will not, however, develop this point here.

6 From TPN's to TSM's

We now give a construction of an equivalent TSM, given a 1-TPN; the differences with Berthomieu and Menasche's [3], [2] *enumerative approach* should be noted.

Let $\mathcal{N} = (P, \mathcal{Q}, Pre, Post, M_0)$ be a bounded TPN and $\mathcal{G} = (S, \mathcal{E})$ its reachability graph. Here, the labeling function $\mathfrak{l} : \mathcal{E} \to \mathcal{Q}$ labels edges of $\mathcal{M}$ by the corresponding net transition. $\mathcal{M}$ contains the reachable markings of $\mathcal{N}$. Set $S := \{s_M : M \in \mathcal{M}\}$, where $M \mapsto s_M$ is injective, $s_o := M_0$ and $\mathcal{E} := \overline{\mathcal{Q}}$.

We associate injectively one clock c_q to every transition q of $\mathcal{N}$; $C(\mathcal{B}) := \{c_q : q \in \mathcal{Q}(\mathcal{N})\}$. Thus, every occurrence e of q will be governed by the same clock c_q. According to the TPN semantics, the clock c_q has to be reset at zero exactly at those physical instants τ that q becomes enabled and such that q was disabled for at least one logical instant immediately before or at τ; this cannot be decided locally with a clock c_e. Thus, consider $M \xrightarrow{q} M'$. Then, for $s_M \xrightarrow{e} s_{M'}$,

$$\begin{aligned} \mathcal{Z}(e) \;\; := \;\; & \{c_{\tilde{q}} : \exists p \in P : \\ & [M(p) - Pre(p,q) \;<\; Pre(p,\tilde{q})] \\ & \wedge\; [(M(p) - Pre(p,q)) + Post(q,p) \;\geqslant\; Pre(p,\tilde{q})]\}\,; \end{aligned}$$

that is, exactly the clocks corresponding to net transitions that are enabled after q *completes* its firing and were *dis*abled after q *started* its firing have to be reset on completing q's firing. Note: This definition is part of those that have to be modified if the assumption of self-concurrency degree 1 is dropped. In Figure 4, the construction is shown in the context of the 1-safe TPN of the left hand side. Markings are given as multi-sets of places; since the net is 1-safe, these multi-sets are just sets.

We can now derive the clock constraints from eft and lft. For any $M \in \mathcal{M}$, $q \in \alpha(M))$ and $e \in f^{-1}(q)$, set

$$\mathcal{D}(e) \;\; := \;\; \left\{ \left(c_{\mathfrak{l}(e)} \in [eft(\mathfrak{l}(e)), lft(\mathfrak{l}(e))]\right) \wedge \bigwedge_{q' \in (\alpha(M) - \{\mathfrak{l}(e)\})} c_{q'} \in [0, lft(q')] \right\}.$$

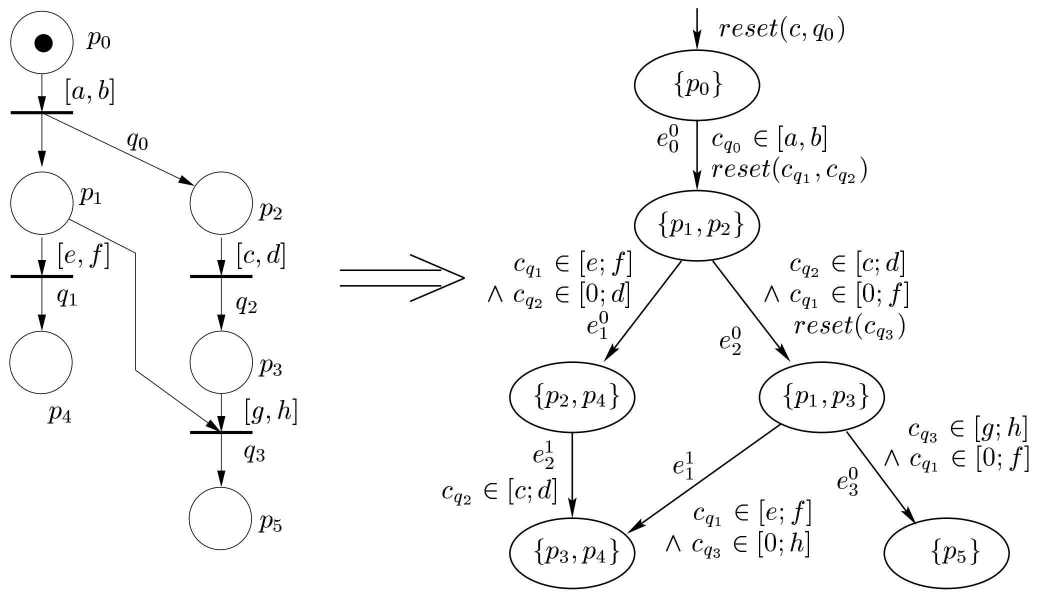

Figure 4: Construction of a TIOSM from a TPN

Thus, $q := \mathfrak{l}(e)$ itself yields an elementary clause on c_q for every such e; the other clauses come from the remaining edges leaving M and reflect the fact that no transition is allowed to delay its firing beyond the latest firing time of a conflicting one. Note that one cannot merge these constraints into a single one because of possible clock resets.

With $\mathcal{M}(\mathcal{N})$ the automaton thus obtained, we obtain inductively the following counterpart of Theorem 1:

Theorem 2 *For every $r^1 = (\sigma^1, \tau^1) \in \mathfrak{sruns}((\mathcal{M}(\mathcal{N}))$ and $r^2 = (\sigma^2, \tau^2) \in \mathfrak{sruns}(\mathcal{N})$ there exist $\overline{r}^1 = (\overline{\sigma}^1, \overline{\tau}^1) \in \mathfrak{sruns}(\mathcal{N})$ and $\overline{r}^2 = (\overline{\sigma}^2, \overline{\tau}^2) \in \mathfrak{sruns}((\mathcal{M}(\mathcal{N}))$ such that*

$$r^1 \equiv_\lambda \bar{r}^1 \quad \textit{and} \quad \overline{r}^2 \equiv_\lambda r^2.$$

Note again that this result can not be extended to runs of $\mathcal{M}$ under weak timing; the assumption of strong timing is essential.

It should also be noted that this translation is different from, and more concise than, the *enumerative method* [3], [2]. In that approach, one considers the graph of classes of states, where the state classes are defined by (untimed) firing sequences leading to a state. Since the same state may be reached by two or more different such sequences, the class graph thus obtained is in general much larger than the reachability graph. By contrast, our method presented here – which can be called the *clock-oriented method* – always yields exactly the reachability graph as the underlying state graph of a timed automaton; this is achieved by making efficient use of the strength given by clocks and clock resets in timed automata. Moreover, going from between the *implicit* clocks of (N1)TPNs to the *explicit* ones in TSMs, one has an entirely natural correspondence, up to the fact that one has to impose strong rather than weak timing on the automaton.

7 Conclusions

TSMs are models that should prove sufficiently general for almost all applications; the extra strength of TA being the possibility of open interval constraints, we doubt that physical systems suitable for automatic verification or testing require distinctions as in Figure 5; no physical process could distinguish an isolated point in time from its neighboring intervals. On the other hand, every TA can be seen as the limit of an appropriately chosen family of TSM, so "proper" TAs can be approximated by TSM. The effective translation given here opens up the possibility to change between TSM and TPN models and to use all the respective techniques and tools of either domain.

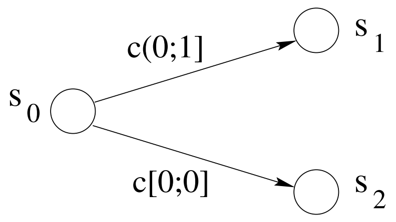

Figure 5: A TA that is not a TSM

Note that TAs such as the one in Figure 5 cannot be modeled by TPN; as the results of Boyer and Diaz [4] show, TPNs are not powerful enough to bisimulate general TAs. For time constraints on *arcs* rather than transitions, the situation is different, see [5] and [4].

The translation of from one model class into the other is natural (up to imposing strong firing) and makes use of the specific power of clocks in either model. Note also that the size of $\mathcal{N}(\mathcal{M})$ is linearly bounded in the product of $|S|$ and the number of elementary clock constrains in $\mathcal{M}$.

Using TIOSMs, one has analysis tool support for *modular* modeling with Time Petri Nets; the translation operations between T(IO)SM and TPN can be – and have been – effectively implemented in the XTIOSM tool [10]; see also [11], [13]).

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.

[2] B. Berthomieu and M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. IEEE Trans. on Software Eng. vol 17 No. 3:259–273, 1991.

[3] B. Berthomieu and M. Menasche. An Enumerative Approach for analyzing time Petri nets. Proceedings IFIP, September 1983. Paris.

[4] M. Boyer and M. Diaz. Non-Equivalence between Time Petri Nets and Time Stream Petri Nets. In: P. Buchholz and M. Silva, editor, *PNPM '99*, pages 198–207, Los Alamitos, 1999. IEEE Computer Society.

[5] A. Cerone and A. Maggiolo Schettini. Time-based expressivity of time Petri nets for system specification. *Theoretical Computer Science*, 216:1–53, 1999.

[6] B. Grahlmann. Combining Finite Automata, Parallel Programs and SDL Using Petri Nets. In: B. Steffen, editor, *TACAS'98*, volume 1384 of *LNCS*, pages 102–117. Springer, 1998.

[7] S. Haar. Occurrence Net Logics. *Fundamenta Informaticae*, 43:105–127, August 2000.

[8] J. Esparza and S. Römer and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In: T. Margaria and B. Steffen, editor, *Proceedings of TACAS '96*, volume 1055 of *LNCS*, pages 87–106, Berlin etc., 1996. Springer Verlag.

[9] J. Lilius. Time processes for time Petri nets. In: P. Azéma and G. Balbo, editors, *Applications and Theory of Petri Nets 1997. 19th Int. Conference, Toulouse, France, June 1997*, volume 1248 of *LNCS*. Springer Verlag, 1997.

[10] L. Kaiser. XTIOSM: A tool for the automatic translation of TPN into TIOSM and vice versa. Internal Report, ENSEM, Vandoeuvre, France.

[11] L. Kaiser. Une méthode de vérification d'interoperabilité temporelle. In: *Colloque francophone RENPAR'9*, pages 207–210, Lausanne, 1997.

[12] L. Kaiser and F. Simonot-Lion. Adaptive Timed Test for Temporal Interoperability Verification. In: *WFCS 2000*, Porto, Portugal, Sept. 2000.

[13] L. Kaiser, F. Simonot-Lion, and O. Koné. Verification Method of Interoperability for real time systems. In: Proc. *SICICA* 2000, Buenos Aires, 2000.

[14] O. Koné and R. Castanet. Conformance with Time Extensions. Technical report, Univérsité de Bordeaux, 1995.

[15] R. Lanotte, A. Maggiolo Schettini, and A. Peron. Timed Cooperating Automata. *Fundamenta Informatica*, 43:153–173, August 2000.

[16] K. McMillan. Using Unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: *4th Workshop on Computer Aided Verification, Montreal 1992*, pages 164–174, 1992.

[17] P. M. Merlin and D. J. Farber. Recoverability of Communication Protocols. *IEEE Transactions of Communications*, 24(9):36–103, September 1976.

[18] J. Toussaint, F. Simonot-Lion, and J. P. Thomesse. Time Constraint Verification Methods Based on Time Petri Nets. 4th Workshop on Future Trends in Distributed Computing Systems. Tunis, Tunisia, October 1997.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399