Timed-Arc Petri Nets vs. Networks of Timed Automata

Jiří Srba*

BRICS**
Department of Computer Science
Aalborg University
Fredrik Bajersvej 7B, 9220 Aalborg East
Denmark
`srba@brics.dk`

Abstract. We establish mutual translations between the classes of 1-safe timed-arc Petri nets (and its extension with testing arcs) and networks of timed automata (and its subclass where every clock used in the guard has to be reset). The presented translations are very tight (up to isomorphism of labelled transition systems with time). This provides a convenient characterization from the theoretical point of view but is not always satisfactory from the practical point of view because of the possible non-polynomial blow up in the size (in the direction from automata to nets). Hence we relax the isomorphism requirement and provide efficient (polynomial time) reductions between networks of timed automata and 1-safe timed-arc Petri nets preserving the answer to the reachability question. This makes our techniques suitable for automatic translation into a format required by tools like UPPAAL and KRONOS. A direct corollary of the presented reductions is a new PSPACE-completeness result for reachability in 1-safe timed-arc Petri nets, reusing the region/zone techniques already developed for timed automata.

1 Introduction

One of the major challenges in theoretical computer science is to establish a precise relationship among a wide range of modelling formalisms available nowadays and to compare their descriptive power and verification capabilities. Recently, various models to describe concurrent systems with real-time features attracted lots of attention. The most wide-spread models include timed automata of Alur and Dill [3] and several timed extensions of Petri nets (see [10, 29] for overviews). Verification tools for timed automata have been developed (including tools like UPPAAL [21], KRONOS [12] and CMC [19]) as well as tools for timed Petri nets (including tools like ROMEO [24] and TINA [6]).

* The author is supported in part by grants ITI-1M0021620808, 1ET-408050503, and GACR 201/03/1161.

** **B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

In this paper we focus on timed-arc Petri nets [7, 17], a model where time (age) is associated with tokens and transitions are labelled by time intervals which restrict the age of tokens that can be used to fire them. We consider the weak (non-urgent) semantics. The reachability problem for general timed-arc Petri nets is undecidable [26], even in the case where tokens in different places are not required to age at the same rate [23]. On the other hand, coverability and boundedness are decidable [25, 1]. We compare the timed-arc Petri net model with timed automata. Unlike the other models of timed Petri nets, timed-arc Petri nets still suffer from a lack of analysis and verification tools and hence automatic translations to already existing tools (e.g. to UPPAAL timed automata) are of a relatively high interest.

Our Contribution. We provide a comparison of the (sub/super)-classes of networks of timed automata (also called concurrent timed automata) and 1-safe timed-arc Petri nets up to isomorphism of labelled transitions systems with time (and hence also up to timed bisimilarity). The usefulness of the presented translations is documented by a PSPACE-completeness result for reachability of 1-safe timed-arc Petri nets where we directly reuse the recent results for reachability in networks of timed automata by Aceto and Laroussinie [2]. At the practical level, we describe polynomial time (and size) translations between the mentioned models up to reachability and we show a technique to reduce the general type of synchronization function in networks of timed automata into the synchronization policy accepted by UPPAAL (handshake synchronization). Last but not least our results provide a natural motivation for the study of the extension of timed-arc Petri net with testing arcs and of the subclass of timed automata where clocks used in guards are mandatorily reset by the same transition.

Related work. Techniques to translate Petri nets extended with time features into equivalent timed automata have not been often studied in the past. Only recently, there has been some focus on translating different variants of timed-transitions Petri nets into timed automata [22, 16, 8]. A work most related to ours is by Cassez and Roux [14]. In their paper they propose (independently of our work) a structural translation of bounded timed-transitions Petri nets into communicating timed automata. Each transition in their model is translated into a single timed automaton and transition firing is supervised by an additional timed automaton. Our model is, however, different from timed-transitions Petri nets (time features are associated to tokens, not to transitions) and we provide a translation that does not rely on additional UPPAAL features like arrays of integers that are used in [14]. As concluded in [11], timed-transitions Petri nets express timed behaviour and timed-arc Petri nets express time behaviour *and* time constraints.

Considering the class we study (timed-arc Petri nets), the only related work we are aware of is by Sifakis and Yovine [27]. They provide a different (non-structural) translation of timed-arc Petri nets (with urgent behaviour) into timed automata (with invariants), hence achieving essentially the result of our Corollary 3, except for the fact that their translation causes an exponential blow up in the size of the timed automaton whereas our translation is structural and can be implemented in polynomial time (and space).

2 Basic Definitions

2.1 Labelled transition systems with time

A *rooted labelled transition system with (real) time* (or simply LTS) is a tuple $T = (S, \mathcal{A}ct, \longrightarrow, s^0)$ where S is a set of *states*, $\mathcal{A}ct$ is a set of *actions* such that $\mathcal{A}ct \cap \mathbb{R}^+ = \emptyset$, $\longrightarrow \subseteq S \times (\mathcal{A}ct \cup \mathbb{R}^+) \times S$ is a *transition relation*, written $s \xrightarrow{a} s'$ for $(s, a, s') \in \longrightarrow$ where $a \in \mathcal{A}ct$, and $s \xrightarrow{\epsilon(r)} s'$ for $(s, r, s') \in \longrightarrow$ where $r \in \mathbb{R}^+$, and $s^0 \in S$ is a distinguished *initial state*.

Hence LTS have two kinds of transitions: the standard ones under the visible actions from $\mathcal{A}ct$ and time-elapsing ones under the actions $\epsilon(r)$, where for all $r \in \mathbb{R}^+$ the symbol $\epsilon(r)$ represents a special action name called a *time delay*.

We write $s \longrightarrow s'$ whenever $s \xrightarrow{a} s'$ for some $a \in \mathcal{A}ct$ or $s \xrightarrow{\epsilon(r)} s'$ for some $r \in \mathbb{R}^+$. Let $\longrightarrow^*$ denote the reflexive and transitive closure of $\longrightarrow$. By $\mathit{Reach}(s^0) \overset{\text{def}}{=} \{s \in S \mid s^0 \longrightarrow^* s\}$ we denote the set of all *reachable states* in T.

Let $T_1 = (S_1, \mathcal{A}ct_1, \longrightarrow_1, s_1^0)$ and $T_2 = (S_2, \mathcal{A}ct_2, \longrightarrow_2, s_2^0)$ be two LTS. We say that T_1 and T_2 are isomorphic (and write $T_1 \cong T_2$) whenever there is a bijection $f : \mathit{Reach}(s_1^0) \to \mathit{Reach}(s_2^0)$ such that for all $a \in \mathcal{A}ct$, $r \in \mathbb{R}^+$ and $s, s' \in \mathit{Reach}(s_1^0)$ it is the case that $s \xrightarrow{a}_1 s'$ iff $f(s) \xrightarrow{a}_2 f(s')$, and $s \xrightarrow{\epsilon(r)}_1 s'$ iff $f(s) \xrightarrow{\epsilon(r)}_2 f(s')$. Hence the reachable parts of the transition systems are identical up to renaming of states.

2.2 Time domains and time intervals

In this paper we consider a continuous time domain, i.e., time values are from the set of nonnegative real numbers $\mathbb{R}^+$. The set of natural numbers (including 0) is denoted by $\mathbb{N}$ and is used in guards.

Remark 1. We can use also discrete time in the time domain and/or rational numbers in guards and this does not influence the results presented in this paper.

The set $\mathcal{I}$ of *time intervals* is defined by the following abstract syntax where a and b range over $\mathbb{N}$ such that $a < b$.

$$I \ ::= \ [a, b] \ \mid \ [a, a] \ \mid \ (a, b] \ \mid \ [a, b) \ \mid (a, b) \ \mid \ [a, \infty) \ \mid \ (a, \infty)$$

Let $I \in \mathcal{I}$. Given a time point $r \in \mathbb{R}^+$, the validity of the expression $r \in I$ is defined in the usual way, e.g., $r \in [a, b)$ iff $a \leq r < b$ and $r \in (a, \infty)$ iff $a < r$.

Remark 2. It is easy to see that any intersection of finitely many time intervals $I_1, \ldots, I_n \in \mathcal{I}$ (denoted by $\cap_{i=1}^n I_i$) is either empty ($\emptyset$) or it belongs to $\mathcal{I}$.

2.3 Timed-arc Petri nets

A *(labelled) timed-arc Petri net* (TAPN) is a tuple $N = (P, T, F, c, \mathcal{A}ct, \lambda)$, where P is a finite set of *places*, T is a finite set of *transitions* such that $T \cap P = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*, $c : F|_{P\times T} \rightarrow \mathcal{I}$ is a *time constraint* assigning a time interval to every arc from a place to a transition, $\mathcal{A}ct$ is a set of *labels* (*actions*), and $\lambda : T \rightarrow \mathcal{A}ct$ a *labelling function*.

We also define ${}^\bullet t \stackrel{\text{def}}{=} \{p \mid (p,t) \in F\}$ and $t^\bullet \stackrel{\text{def}}{=} \{p \mid (t,p) \in F\}$. Let $N = (P, T, F, c, \mathcal{A}ct, \lambda)$ be a TAPN. A *marking* M on the net N is a function $M\colon P \rightarrow \mathcal{B}(\mathbb{R}^+)$ where $\mathcal{B}(\mathbb{R}^+)$ denotes the set of finite multisets on $\mathbb{R}^+$. Each place is thus assigned a certain number of tokens, and each token is annotated with a real number (*age*). Let $B \in \mathcal{B}(\mathbb{R}^+)$ and $a \in \mathbb{R}^+$. We define $B + a$ in such a way that we add the value a to every element of B, i.e., $B+a \stackrel{\text{def}}{=} \{b+a \mid b \in B\}$. As *initial markings* we allow only markings with all tokens of age 0. A *marked TAPN* is a pair (N, M_0) where N is a timed-arc Petri net and M_0 is an initial marking.

Let us now define the dynamics of TAPN. We introduce two types of transition rules: *firing* of a transition and *time-elapsing*.

Let $N = (P, T, F, c, \mathcal{A}ct, \lambda)$ be a TAPN, M a marking and $t \in T$.

- We say that t is *enabled* by M iff $\forall p \in {}^\bullet t.\ \exists x \in M(p).\ x \in c(p,t)$.
- If t is enabled by M then it can *fire*, producing a marking M' such that:

$$\forall p \in P.\ M'(p) = \Big(M(p) \smallsetminus C^-(p,t)\Big) \cup C^+(t,p)$$

  where C^- and C^+ are chosen to satisfy the following equations (note that there may be more possibilities and that all the operations are on multisets):

$$C^-(p,t) = \begin{cases} \{x\} & \text{if } p \in {}^\bullet t \text{ s.t. } x \in M(p) \wedge x \in c(p,t) \\ \emptyset & \text{otherwise} \end{cases}$$

$$C^+(t,p) = \begin{cases} \{0\} & \text{if } p \in t^\bullet \\ \emptyset & \text{otherwise.} \end{cases}$$

  Then we write $M[t\rangle M'$. Note that the new tokens added to the places in $t^\bullet$ are of the initial age 0.
- We define a *time-elapsing* transition $\epsilon(r)$, for $r \in \mathbb{R}^+$, as follows:

$$M[\epsilon(r)\rangle M' \quad \text{iff} \quad \forall p \in P.\ M'(p) = M(p) + r.$$

A marked TAPN (N, M_0) where $N = (P, T, F, c, \mathcal{A}ct, \lambda)$ generates a LTS

$$T(N, M_0) \stackrel{\text{def}}{=} (P \rightarrow \mathcal{B}(\mathbb{R}^+), \mathcal{A}ct, \longrightarrow, M_0)$$

where states are markings on N, and the transition relation $\longrightarrow$ is defined as follows:

$$M \xrightarrow{a} M' \text{ whenever } M[t\rangle M' \text{ for some } t \in T \text{ such that } \lambda(t) = a$$
$$M \xrightarrow{\epsilon(r)} M' \text{ whenever } M[\epsilon(r)\rangle M'.$$

In standard P/T Petri nets there is a simple construction to ensure that a transition can be fired only if a token is present in a certain place, without removing the token. This is done by adding two arcs: one from the place to the transition and one in the opposite direction. A similar construction, however, does not work in TAPN, as consuming a (timed) token resets its age. In order to recover this possibility we shall explicitly add so called *testing arcs*. Testing arcs (called *read arcs* in e.g. [28, 13]) were also investigated in connection with partial order semantics for (untimed) P/T nets.

A *timed-arc Petri net with testing arcs* is a tuple $N = (P, T, F, c, \mathcal{A}ct, \lambda, F{*}, c{*})$ such that $(P, T, F, c, \mathcal{A}ct, \lambda)$ is a timed-arc Petri net, $F{*} \subseteq P \times T$ is a set of *testing arcs*, and $c{*} : F{*} \to \mathcal{I}$ is a function which assigns a time interval to every testing arc from $F{*}$. We define ${}^{*}t \stackrel{\text{def}}{=} \{p \mid (p, t) \in F{*}\}$.

The dynamics of TAPN with testing arcs is defined as in the case of TAPN with the only difference that for a transition t to be enabled, it has to satisfy an extra condition, namely

$$\forall p \in {}^{*}t.\ \exists x \in M(p).\ x \in c{*}(p, t).$$

In other words, a necessary condition for a transition to fire is that all places which are connected via testing arcs to the transition contain a token satisfying the constraint on the testing arc. The transition can then fire according to the rules defined above, which means that the testing arcs do not consume any tokens and hence do not influence their age.

A *1-safe marking* is a marking M having at most one token in every place, i.e., $|M(p)| \leq 1$ for all $p \in P$. A *1-safe marked TAPN* is a marked TAPN (N, M_0) where all markings from $Reach(M_0)$ are 1-safe markings. In case of 1-safe nets we will require that $F \cap F{*} = \emptyset$. This is without loss of generality as the conditions on testing arcs can be inserted (by Remark 2) onto standard arcs whenever we know that every place contains at most one token.

We shall use the following abbreviations: TAPN for timed-arc Petri nets; TAPN* for timed-arc Petri nets with testing arcs; 1-TAPN for 1-safe timed-arc Petri nets; and 1-TAPN* for 1-safe timed-arc Petri nets with testing arcs. The size of (N, M_0) where $N = (P, T, F, c, \mathcal{A}ct, \lambda, F{*}, c{*})$ is the size of N (the size of the description of N) plus the size of M_0, formally $size(N, M_0) \stackrel{\text{def}}{=} |P| + |T| + |F| + |\mathcal{A}ct| + |F{*}| + \sum_{p \in P} |M_0(p)|$.

2.4 Concurrent timed automata

Let C be a finite set of *clocks*. A *(time) valuation* of clocks from C is a function $v : C \to \mathbb{R}^+$. Let v be a valuation and $r \in \mathbb{R}^+$. We define a valuation $v + r : C \to \mathbb{R}^+$ by $(v + r)(x) \stackrel{\text{def}}{=} v(x) + r$ for every $x \in C$. For every set $R \subseteq C$ we define a valuation $v[R := 0] : C \to \mathbb{R}^+$ by $v[R := 0](x) \stackrel{\text{def}}{=} v(x)$ for $x \in C \smallsetminus R$ and $v[R := 0](x) \stackrel{\text{def}}{=} 0$ for $x \in R$. A *clock guard* is a partial function $g : C \hookrightarrow \mathcal{I}$ assigning a time interval to selected clocks. We say that a valuation v satisfies a guard g (written $v \models g$) iff $v(x) \in g(x)$ for all $x \in dom(g)$.

Remark 3. The definition of a clock guard given above enables to encode also boolean combinations of clock constraints in such a way that conjunction of several guards for a certain clock is replaced by intersection using Remark 2 and disjunction is replaced by multiple edges (see e.g. [5]). We also do not consider difference constraints since from the expressive point of view there are techniques for their replacement with simple constraints [5] and because of the well accepted observation that difference constraints are not crucial for modelling real-life systems [9].

A *timed automaton* (TA) is a tuple $A = (S, \mathcal{A}ct, C, \longrightarrow, s^0)$ where S is a finite set of *control states*, $\mathcal{A}ct$ is a finite set *actions*, C is a finite set of *clocks*, $\longrightarrow \subseteq S \times (C \hookrightarrow \mathcal{I}) \times \mathcal{A}ct \times 2^C \times S$ is a finite *transition relation* written $s \xrightarrow{g,a,R} s'$ for $(s, g, a, R, s') \in \longrightarrow$, and $s^0 \in S$ is an *initial* control state.

A *configuration* of a timed automaton A is a pair (s, v) where s is a control state ($s \in S$) and v is a time valuation on C ($v : C \to \mathbb{R}^+$). An *initial configuration* of A is (s^0, v^0) such that $v^0(x) \overset{\text{def}}{=} 0$ for all $x \in C$. A given timed automaton $A = (S, \mathcal{A}ct, C, \longrightarrow, s^0)$ determines a LTS

$$T(A) \overset{\text{def}}{=} (S \times (C \to \mathbb{R}^+), \mathcal{A}ct, \longrightarrow, (s^0, v^0))$$

where states are configuration of A and the transition relation $\longrightarrow$ is defined by

$$(s, v) \xrightarrow{a} (s', v[R := 0]) \text{ whenever there is a transition } s \xrightarrow{g,a,R} s' \text{ in } A \text{ s.t. } v \models g$$
$$(s, v) \xrightarrow{\epsilon(r)} (s, v + r) \qquad \text{for all } r \in \mathbb{R}^+.$$

In modelling real-time systems using timed automata, we often design several subcomponents and then define the composed behaviour of the whole system. This is usually done by parallel composition with a certain communication scheme (see e.g. the tools UPPAAL [21] and KRONOS [12]). Assume that the system consists of n concurrent components. As suggested e.g. in [2] a general synchronization scheme in the style of Arnold-Nivat [4] can be described by a so called *synchronization function* $\phi : (\mathcal{A}ct \cup \{\bullet\})^n \hookrightarrow \mathcal{A}ct$ which is a partial function where $\bullet$ denotes a distinguished symbol of *inactivity* such that $\bullet \notin \mathcal{A}ct$. The components different from $\bullet$ are called *active* components and we require that $(\bullet, \bullet, \ldots, \bullet) \notin dom(\phi)$, i.e., at least one component in every synchronization tuple is active. By $|\phi|$ we understand the cardinality of the set $dom(\phi)$ and by $width(\phi)$ we denote the maximum number of active components over all tuples from $dom(\phi)$.

Let $A_1, \ldots, A_n$ be timed automata where (for all i, $1 \le i \le n$) $A_i = (S_i, \mathcal{A}ct, C, \longrightarrow_i, s_i^0)$ and where $\mathcal{A}ct$ and C are fixed sets of actions and clocks, respectively. Let ϕ be a synchronization function. A *concurrent timed automaton* (CTA) with a synchronization function ϕ is a parallel composition of $A_1, \ldots, A_n$ denoted by $A = (A_1 \| \cdots \| A_n)_\phi$.

A CTA A determines an LTS

$$T(A) \overset{\text{def}}{=} (S_1 \times \cdots \times S_n \times (C \to \mathbb{R}^+), \mathcal{A}ct, \longrightarrow, (s_1^0, \ldots, s_n^0, v^0))$$

where states are Cartesian products of the control states of the individual automata together with a clock valuation, and the transition relation $\longrightarrow$ is defined by

– $(s_1,\ldots,s_n,v) \xrightarrow{a} (s'_1,\ldots,s'_n,v')$ whenever there is a tuple $(a_1,\ldots,a_n) \in \mathit{dom}(\phi)$ such that:
  - for all i, $1 \le i \le n$, with $a_i = \bullet$ we have $s'_i = s_i$, and we set $R_i \overset{\text{def}}{=} \emptyset$
  - for all i, $1 \le i \le n$, with $a_i \in \mathcal{A}ct$ we have (in A_i) a transition rule $s_i \xrightarrow{g_i,a_i,R_i} s'_i$ s.t. $v \models g_i$
  - $v' = v[R := 0]$ where R is defined by $R \overset{\text{def}}{=} \cup_{i=1}^{n} R_i$,
  - $a = \phi(a_1,\ldots,a_n)$

– $(s_1,\ldots,s_n,v) \xrightarrow{\epsilon(r)} (s_1,\ldots,s_n,v+r)$ for all $r \in \mathbb{R}^+$.

This means that the composition of timed automata can perform a synchronization step whenever the individual automata can perform transitions labelled according to the function ϕ. Moreover, if some automaton resets a clock, the clock is also reset in the concurrent timed automaton. The size of a concurrent automaton A is the sum of the sizes of all its components A_i plus the size of the synchronization function ϕ, formally $\mathit{size}(A) \overset{\text{def}}{=} |\phi| + \sum_{i=1}^{n} |A_i|$ where $|A_i|$ for $A_i = (S_i, \mathcal{A}ct, C, \longrightarrow_i, s_i^0)$ is equal to $|S_i| + |\mathcal{A}ct| + |C| + |\longrightarrow_i|$.

We now define subclasses of TA and CTA called *mandatory reset* TA (mrTA) and *mandatory reset* CTA (mrCTA). The intuition is that every clock which is tested in a guard while performing a transition must be mandatorily reset on that transition. Formally, a mandatory reset TA is a TA such that every transition $s \xrightarrow{g,a,R} s'$ in the automaton satisfies $\mathit{dom}(g) \subseteq R$. A CTA is called a mandatory reset CTA iff each of its components is a mrTA.

Remark 4. The idea of mandatory reset in the case of one clock only was considered by Laroussinie, Markey and Schnoebelen in [20]. Their definition forms a natural subclass of timed automata and we will provide further justification of this notion in what follows.

3 From Automata to Nets

We shall now demonstrate that the class of LTS generated by CTA is contained (up to isomorphism) in the class of LTS generated by 1-TAPN*, i.e., we present a construction that for a given CTA A algorithmically defines a marked 1-TAPN* (N, M_0) such that $T(A) \cong T(N, M_0)$.

Let $\{A_i = (S_i, \mathcal{A}ct, C, \longrightarrow_i, s_i^0)\}_{i=1}^{n}$ be a collection of TA and let ϕ be an arbitrary synchronization function. Without loss of generality we assume that the sets of control states S_i are pairwise disjoint for all i, $1 \le i \le n$, and that the function ϕ satisfies that whenever $(a_1,\ldots,a_n) \in \mathit{dom}(\phi)$ then for each a_i different from $\bullet$ there exists at least one transition rule in A_i labelled with the action a_i.

Remark 5. Without loss of generality we also assume that every TA A_i contains a distinguished clock $\mathit{delay}_i \in C$ which is never used in any guard of any timed automaton but is always reset in all transition rules of the automaton A_i. This means that the clock delay_i measures how much time has elapsed (in the i'th component) from the last occurrence of a transition labelled by an action from $\mathcal{A}ct$. These additional clocks do not influence the behaviour of the concurrent automaton A and are considered only for technical reasons in order to establish isomorphism of the respective labelled transition systems. They are not necessary if we were interested in weaker equivalence notions (e.g. in timed bisimilarity).

We now consider a CTA $A = (A_1 \| \cdots \| A_n)_\phi$ with the initial configuration $(s_1^0, \ldots, s_n^0, v^0)$ where $v^0(x) \overset{\text{def}}{=} 0$ for all $x \in C$, and with the special clocks delay_i according to Remark 5. We shall construct a 1-TAPN* $N \overset{\text{def}}{=} (P, T, F, c, \mathcal{A}ct, \lambda, F*, c*)$ with an initial marking M_0 such that it generates an LTS isomorphic to $T(A)$. The intuition is that every tuple $(a_1, \ldots, a_n)$ from $\mathit{dom}(\phi)$ defines a set of transitions in the net, which simulate the effect of the synchronization function. The set of transitions enumerates all possible combinations of transition rules available in the components of the CTA and labelled by the corresponding actions. Places in the net represent control states and there are also special places designed for storing tokens representing clocks. An extra place called e (which will never become marked) is added for a technical convenience.

$P \overset{\text{def}}{=} C \cup \{s \mid s \in \cup_{i=1}^n S_i\} \cup \{e\}$
$T \overset{\text{def}}{=} \{t_{(a_1,\ldots,a_n),(w_1,\ldots,w_n)} \mid (a_1, \ldots, a_n) \in \mathit{dom}(\phi)$ and for all i, $1 \le i \le n$, $w_i = \bullet$ if $a_i = \bullet$; otherwise $w_i = (s_i, g_i, a_i, R_i, s_i')$ whenever there is a transition rule $s_i \xrightarrow{g_i, a_i, R_i}_i s_i'$ in $A_i\}$

For every transition $t_{(a_1,\ldots,a_n),(w_1,\ldots,w_n)} \in T$ we define $\lambda(t_{(a_1,\ldots,a_n),(w_1,\ldots,w_n)}) \overset{\text{def}}{=} \phi(a_1, \ldots, a_n)$. The set of arcs associated with every transition from T is given as follows. Let $t = t_{(a_1,\ldots,a_n),(w_1,\ldots,w_n)} \in T$ where every w_i is either $\bullet$ or of the form $(s_i, g_i, a_i, R_i, s_i')$ for all i, $1 \le i \le n$. We define a set $J \subseteq \{1, \ldots, n\}$ of active components of t by $i \in J$ iff $w_i \neq \bullet$, and the set of all reset clocks by $R \overset{\text{def}}{=} \cup_{i \in J} R_i$. For each clock $x \in C$ such that there is at least one g_i where $x \in \mathit{dom}(g_i)$ we also define the combined guard

$$I_x \overset{\text{def}}{=} \bigcap_{i \in J \,\wedge\, x \in \mathit{dom}(g_i)} g_i(x)$$

which expresses a combined requirement (time interval) of all active components of A on the clock x. In the case that these requirements are not consistent, we have $I_x = \emptyset$ according to Remark 2. Let us now construct the set of arcs of N incident with t.

- For every w_i, $i \in J$, we add the following arcs: $(s_i, t) \in F$ with $c(s_i, t) = [0, \infty)$ and $(t, s_i') \in F$.

(The intuition is that a token in a place s_i means that A_i is in the control state s_i and the defined arcs represent the change of control states of the active automata; the age of the tokens is irrelevant here.)

- For every clock $x \in C$ with at least one g_i, $i \in J$, such that $x \in dom(g_i)$ we add the following arcs (the age of the token in place x represents the corresponding clock value):
  - $(x, t) \in F$ with $c(x, t) = I_x$ and $(t, x) \in F$ whenever $I_x \neq \emptyset$ and $x \in R$ *(The clock x is used in a guard and reset afterwards.)*
  - $(x, t) \in F*$ with $c*(x, t) = I_x$ whenever $I_x \neq \emptyset$ and $x \notin R$ *(The clock x is used in a guard but not reset.)*
  - $(e, t) \in F$ with $c(e, t) = [0, \infty)$ whenever $I_x = \emptyset$ *(If the guards for x are inconsistent, t is disabled: the place e will never become marked.)*
- For every clock $x \in R$ such that there is no g_i, $i \in J$, where $x \in dom(g_i)$ we add the arcs: $(x, t) \in F$ with $c(x, t) = [0, \infty)$ and $(t, x) \in F$. *(The clock x is not used in any guard but it is reset.)*

The construction is schematically depicted in Figure 1 using the standard Petri net notation (testing arcs are drawn by dashed arrows).

For every configuration $(s_1, \ldots, s_n, v)$ of A reachable from its initial state we define a corresponding marking $M_{(s_1,\ldots,s_n,v)}$ in N by

$$M_{(s_1,\ldots,s_n,v)}(p) \overset{\text{def}}{=} \begin{cases} \{v(delay_i)\} & \text{if } p = s_i \text{ for some } i, 1 \leq i \leq n \\ \{v(p)\} & \text{if } p \in C \\ \emptyset & \text{otherwise.} \end{cases}$$

The initial marking M_0 of N is then given by $M_0 \overset{\text{def}}{=} M_{(s_1^0,\ldots,s_n^0,v^0)}$. It is easy to see that (N, M_0) is a 1-safe net.

Theorem 1. *Every CTA can be transformed into 1-TAPN* preserving isomorphism of LTS.*

Proof. It can be verified that for any reachable configuration $(s_1, \ldots, s_n, v)$ of A whenever we have $(s_1, \ldots, s_n, v) \xrightarrow{a} (s'_1, \ldots, s'_n, v')$ then also $M_{(s_1,\ldots,s_n,v)} \xrightarrow{a} M_{(s'_1,\ldots,s'_n,v')}$ and vice versa (any transition from $M_{(s_1,\ldots,s_n,v)}$ corresponds to a transition from $(s_1, \ldots, s_n, v)$). The same holds also for time-elapsing transitions under the actions $\epsilon(r)$ where $r \in \mathbb{R}^+$. $\square$

Corollary 1. *Every mrCTA can be transformed into 1-TAPN preserving isomorphism of LTS.*

Proof. By inspecting the translation presented above we notice that no testing arcs are used, provided that the CTA A is with mandatory reset. $\square$

Remark 6. In general the translation from CTA to 1-TAPN* (and from mrCTA to 1-TAPN) produces more than an exponential blow up in the size of the 1-TAPN* (caused by the number of transitions; in general this number can be

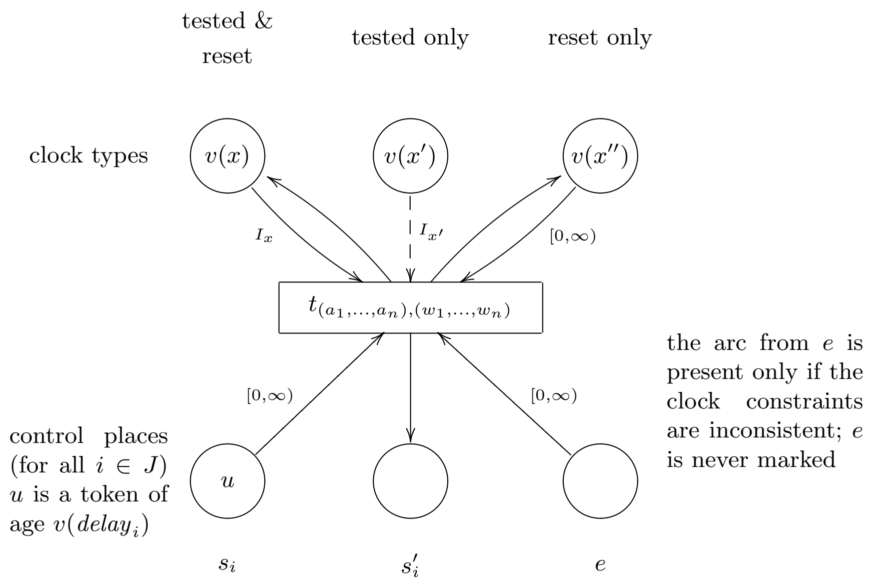

tested & reset
tested only
reset only
clock types
$v(x)$
$v(x')$
$v(x'')$
I_x
$I_{x'}$
$[0,\infty)$
$t_{(a_1,\ldots,a_n),(w_1,\ldots,w_n)}$
the arc from e is present only if the clock constraints are inconsistent; e is never marked
$[0,\infty)$
$[0,\infty)$
control places (for all $i \in J$) u is a token of age $v(delay_i)$
u
s_i
s'_i
e

Fig. 1. Schematic construction of the net N

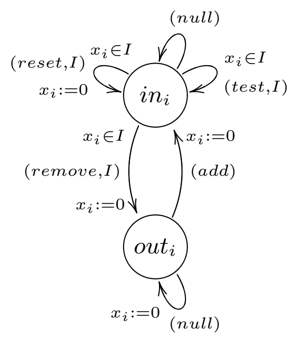
$k^{\Omega(width(\phi))}$ where $k \overset{\text{def}}{=} \max_{1 \le i \le n} | \longrightarrow_i |$, which is $k^{\Omega(n)}$ if ϕ is of the maximal width n). However, considering synchronization functions of constant width (and this is often a common practice — e.g. UPPAAL uses only synchronization of width 2), the translation defines a 1-TAPN* of polynomial size with regard to the size of the CTA.

Corollary 2. *Every CTA resp. mrCTA where the width of the synchronization function is constant can be transformed in polynomial time into a 1-TAPN* resp. 1-TAPN (of polynomial size) preserving isomorphism of LTS.*

4 From Nets to Automata

In this section we present a structural translation from 1-TAPN* to CTA preserving isomorphism of LTS. Let $N = (P, T, F, c, \mathcal{A}ct, \lambda, F*, c*)$ be 1-TAPN* and M_0 its initial marking.

Remark 7. Without loss of generality assume that $P = \{p_1, \ldots, p_n, \mathit{fired}\}$ where $p_1, \ldots, p_n$ are the standard places of the net and *fired* is a newly added place together with the following arcs (for all $t \in T$): $(\mathit{fired}, t) \in F$ and $(t, \mathit{fired}) \in F$ such that $c(\mathit{fired}, t) = [0, \infty)$ and $M_0(\mathit{fired}) = \{0\}$. The intuition is that the place *fired* will always contain exactly one token (representing the time elapsed from the most recent transition firing). This token has no influence on the behaviour of the net.



$$a_i \overset{\text{def}}{=} \begin{cases} (remove, c(p_i,t)) & \text{if } p_i \in {}^\bullet t \smallsetminus t^\bullet \\ (reset, c(p_i,t)) & \text{if } p_i \in {}^\bullet t \cap t^\bullet \\ (test, c{*}(p_i,t)) & \text{if } p_i \in {}^* t \\ (add) & \text{if } p_i \in t^\bullet \smallsetminus {}^\bullet t \\ (null) & \text{otherwise.} \end{cases}$$

Fig. 2. Timed automaton A_i and definition of the synchronization tuples

We shall construct a CTA A (having n parallel components) such that $T(A)$ is isomorphic to $T(N, M_0)$. Let us define a set *Interv* of all time intervals[1] that appear as a time constraint in N, i.e., $\mathit{Interv} \overset{\text{def}}{=} \mathit{range}(c) \cup \mathit{range}(c{*})$. For all i, $1 \leq i \leq n$, we define a timed automaton A_i representing a place of the net N and storing the information whether a token is present in the place p_i (control state in_i) or not (control state out_i), and what is its age (the age will be stored in a clock x_i). In this reduction every transition of the net N will correspond to a certain tuple in the synchronization function ϕ. Let $\mathcal{A}ct \overset{\text{def}}{=} \{(remove, I) \mid I \in \mathit{Interv}\} \cup \{(reset, I) \mid I \in \mathit{Interv}\} \cup \{(test, I) \mid I \in \mathit{Interv}\} \cup \{(add), (null)\}$ and let $C \overset{\text{def}}{=} \{x_1, \dots, x_n\}$. We define $A_i \overset{\text{def}}{=} (S_i, \mathcal{A}ct, C, \longrightarrow_i, s_i^0)$ such that

- $S_i \overset{\text{def}}{=} \{in_i, out_i\}$,
- for all $I \in \mathit{Interv}$ we have the following transitions:

$$\begin{array}{lll} in_i \overset{g,(remove,I),\{x_i\}}{\longrightarrow} out_i & in_i \overset{g,(reset,I),\{x_i\}}{\longrightarrow} in_i & in_i \overset{g,(test,I),\emptyset}{\longrightarrow} in_i \\ out_i \overset{g',(add),\{x_i\}}{\longrightarrow} in_i & in_i \overset{g',(null),\emptyset}{\longrightarrow} in_i & out_i \overset{g',(null),\{x_i\}}{\longrightarrow} out_i \end{array}$$

  such that $g(x_i) \overset{\text{def}}{=} I$ and $g(x_j) \overset{\text{def}}{=} \mathit{undef}$ for all j, $1 \leq j \neq i \leq n$; and $g'(x_j) \overset{\text{def}}{=} \mathit{undef}$ for all j, $1 \leq j \leq n$,
- $s_i^0 \overset{\text{def}}{=} out_i$ if $M_0(p_i) = \emptyset$, and $s_i^0 \overset{\text{def}}{=} in_i$ otherwise.

A picture of the timed automaton A_i is depicted in Figure 2. Let us now consider a CTA $A \overset{\text{def}}{=} (A_1 \| \cdots \| A_n)_\phi$ where for every transition $t \in T$ we define $\phi(a_1, \dots, a_n) \overset{\text{def}}{=} \lambda(t)$ such that the tuple $(a_1, \dots, a_n)$ is given in Figure 2. For the remaining tuples not defined in Figure 2, the partial function ϕ is undefined. Note that because N is a 1-safe net, we can freely assume that ${}^*t \cap t^\bullet = \emptyset$.

[1] For technical convenience, only one *Interv* set is defined. The construction can be further optimized by considering a separate Interv_p (for every place p of the net) containing only the relevant intervals.

Remark 8. In case that a certain place is not involved in firing the transition t, we use the action $(null)$ in the synchronization tuple. The reason is again that we aim at proving isomorphism of the labelled transition systems: if a token is present in such a place, performing the action $(null)$ does not have any influence; if there is no token in the place then the action $(null)$ resets the corresponding clock x_i and hence we know that all the empty places have the clock x_i set to the age of the token in the Petri net place *fired* according to Remark 7. If we aimed at relating the transition systems up to e.g. timed bisimilarity, the $(null)$ actions can be replaced with $\bullet$.

Theorem 2. *Every 1-TAPN* can be transformed in polynomial time into a CTA (of polynomial size) preserving isomorphism of LTS.*

Proof. Every marking M reachable in (N, M_0) defines a unique configuration $(s_1, \ldots, s_n, v)$ of the CTA A such that

- $s_i \overset{\text{def}}{=} in_i$ and $v(x_i) \overset{\text{def}}{=} r$ if $M(p_i) = \{r\}$
- $s_i \overset{\text{def}}{=} out_i$ and $v(x_i) \overset{\text{def}}{=} r$ if $M(p_i) = \emptyset$ and $M(\mathit{fired}) = \{r\}$.

It is now easy to verify that every transition (including time elapsing) from M in the net can be matched by a transition under the same action from $(s_1, \ldots, s_n, v)$ such that the mapping defined above is preserved and vice versa. $\square$

Corollary 3. *Every 1-TAPN can be transformed in polynomial time into a mrCTA (of polynomial size) preserving isomorphism of LTS.*

Proof. Observe that the construction translates 1-TAPN into mandatory reset CTA. $\square$

Theorem 3. *Reachability for 1-TAPN* (and 1-TAPN) is PSPACE-complete.*

Proof. The hardness of the problem follows from PSPACE-hardness of the reachability problem for (untimed) 1-safe Petri nets [15]. The containment follows from Theorem 2 and from the fact that reachability is decidable in PSPACE for CTA [2]. $\square$

5 Reducing the Width of Synchronization Function

So far we have characterized the correspondence between the classes of automata and nets up to isomorphism. This has established a precise relationship w.r.t. to expressiveness, however, when transforming automata to nets, the reduction does not work in polynomial time. Moreover, when transforming nets to automata (in polynomial time), we use a synchronization function of width n, hence the reduction does not provide a direct way to verify problems for 1-TAPN* (and 1-TAPN) by means of UPPAAL, which allows for handshake synchronization (width 2) only. Even the possibility of broadcast communication does not seem to help in this case.

Most of the practical approaches to verification of timed systems focus on reachability. In this section we will hence describe a polynomial time reachability preserving reduction from CTA with arbitrary synchronization function into CTA with synchronization function of width 2.

Let $A = (A_1 \| \cdots \| A_n)_\phi$ be a CTA with an arbitrary synchronization function ϕ such that $A_i = (S_i, \mathcal{A}ct, C, \longrightarrow_i, s_i^0)$.

Remark 9. Without loss of generality we can assume that the CTA A contains a distinguished clock $delay \in C$ which is never used in any guard of any A_i but is always reset in all transition rules of every single automaton. This means that the clock *delay* measures how much time has elapsed from the last occurrence of some transition labelled by an action from $\mathcal{A}ct$, but does not influence the behaviour of the concurrent automaton. We also assume that there are at least two active components in every synchronization tuple from $dom(\phi)$ (if not we can add an additional "dummy" component).

We will construct a CTA A' with a synchronization function ϕ' of width 2 such that reachability in A is reducible in polynomial time into reachability in A'.

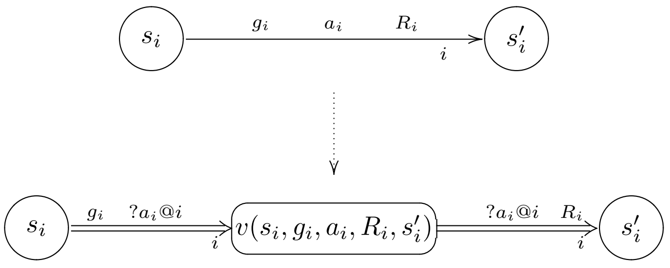
First, we define timed automata A_i' (small modifications of A_i where new intermediate states are inserted for every transition in A_i). Let the new sets of actions and clocks be defined by $\mathcal{A}ct' \overset{\text{def}}{=} \mathcal{A}ct \cup \{!a@i, ?a@i \mid a \in \mathcal{A}ct, 1 \le i \le n\} \cup \{\tau\}$ and $C' \overset{\text{def}}{=} C \cup \{z\}$ where τ is a fresh action and z is a fresh clock. The intuition is that actions of the form $!a@i$ and $?a@i$ (or $!a_i@i$ and $?a_i@i$) where $a, a_i \in \mathcal{A}ct$ are designed for synchronization with the i'th component. For all i, $1 \le i \le n$, the automata A_i' are defined by

$$A_i' \overset{\text{def}}{=} (S_i \cup \{v(s_i, g_i, a_i, R_i, s_i') \mid (s_i, g_i, a_i, R_i, s_i') \in \longrightarrow_i\}, \mathcal{A}ct', C', \Longrightarrow_i, s_i^0)$$

such that $v(s_i, g_i, a_i, R_i, s_i')$ are newly added states and the transition relation $\Longrightarrow_i$ is given by:

- $s_i \overset{g_i, ?a_i@i, \emptyset}{\Longrightarrow}_i v(s_i, g_i, a_i, R_i, s_i')$ and
- $v(s_i, g_i, a_i, R_i, s_i') \overset{g', ?a_i@i, R_i}{\Longrightarrow}_i s_i'$ where $g'(x) \overset{\text{def}}{=} undef$ for all $x \in C'$

for all $(s_i, g_i, a_i, R_i, s_i') \in \longrightarrow_i$. The following picture illustrates the transformation.



For a given tuple $(a_1, \dots, a_n) \in (\mathcal{A}ct \cup \{\bullet\})^n$ let $J(a_1, \dots, a_n) \overset{\text{def}}{=} \{i \in \{1, \dots, n\} \mid a_i \neq \bullet\}$ be the set of active components. We define a new parallel component of A', a timed automaton A'_{n+1} given by

$$A'_{n+1} \overset{\text{def}}{=} (S'_{n+1}, \mathcal{A}ct', C', \Longrightarrow_{n+1}, \mathit{init})$$

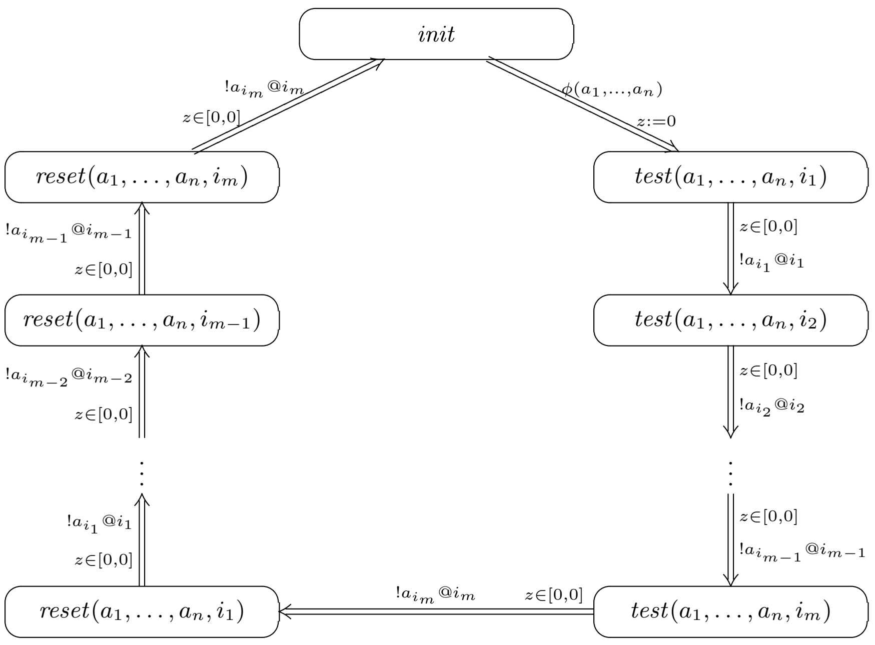
where $S'_{n+1} \overset{\text{def}}{=} \{\ \mathit{test}(a_1, \dots, a_n, i),\ \mathit{reset}(a_1, \dots, a_n, i) \mid (a_1, \dots, a_n) \in \mathit{dom}(\phi),$ $i \in J(a_1, \dots, a_n)\} \cup \{\mathit{init}\}$ with the following transitions for all $(a_1, \dots, a_n) \in \mathit{dom}(\phi)$ (assume that $J(a_1, \dots, a_n) = \{i_1, \dots, i_m\}$ such that $i_1 < i_2 < \dots < i_m$):

- $\mathit{init} \overset{g',a,\{z\}}{\Longrightarrow}_{n+1} \mathit{test}(a_1, \dots, a_n, i_1)$ where $a = \phi(a_1, \dots, a_n)$
- $\mathit{test}(a_1, \dots, a_n, i_\ell) \overset{g_z,!a_{i_\ell}@i_\ell,\emptyset}{\Longrightarrow}_{n+1} \mathit{test}(a_1, \dots, a_n, i_{\ell+1})$ for all ℓ, $1 \leq \ell < m$
- $\mathit{test}(a_1, \dots, a_n, i_m) \overset{g_z,!a_{i_m}@i_m,\emptyset}{\Longrightarrow}_{n+1} \mathit{reset}(a_1, \dots, a_n, i_1)$
- $\mathit{reset}(a_1, \dots, a_n, i_\ell) \overset{g_z,!a_{i_\ell}@i_\ell,\emptyset}{\Longrightarrow}_{n+1} \mathit{reset}(a_1, \dots, a_n, i_{\ell+1})$ for all ℓ, $1 \leq \ell < m$
- $\mathit{reset}(a_1, \dots, a_n, i_m) \overset{g_z,!a_{i_m}@i_m,\emptyset}{\Longrightarrow}_{n+1} \mathit{init}$

where g' is the empty guard ($g'(x) \overset{\text{def}}{=} \mathit{undef}$ for all $x \in C'$), and $g_z(z) \overset{\text{def}}{=} [0, 0]$ and $g_z(x) \overset{\text{def}}{=} \mathit{undef}$ for all $x \in C' \smallsetminus \{z\}$. It first decides which tuple from the domain of ϕ is going to be synchronized upon and then performs twice the sequence of actions $!a_{i_1}@i_1, \dots, !a_{i_m}@i_m$ and becomes init again. The intuition is that the action $!a_{i_\ell}@i_\ell$ can synchronize only with the corresponding action $?a_{i_\ell}@i_\ell$ in the component i_ℓ. Because of the guard g_z no time elapsing steps are possible during the pairwise synchronization, otherwise the whole system gets stuck (and hence cannot reach the requested configuration). During the first sequence of actions $!a_{i_1}@i_1, \dots, !a_{i_m}@i_m$ the guards of the active components are consecutively verified and in the second round the requested clocks are reset. (Note that it is not possible to do both guard verification and resetting of clocks in one run as clocks reset in some earlier synchronized components can still be used in guards later on.) A graphical representation of the automaton A_{n+1} is depicted in Figure 3 (only one loop for a particular $(a_1, \dots, a_n) \in \mathit{dom}(\phi)$ is included). The synchronization function ϕ' is defined exactly for the following tuples:

- $\phi'(\bullet, \dots, \bullet, a) \overset{\text{def}}{=} a$ for all $a \in \mathcal{A}ct$
  (only the last component can make moves without synchronizing with other components; in the initial state init it selects an element $(a_1, \dots, a_n)$ from $\mathit{dom}(\phi)$ to be used)
- $\phi'(\bullet, \dots, \bullet, ?a@i, \bullet, \dots, \bullet, !a@i) \overset{\text{def}}{=} \tau$ for all $a \in \mathcal{A}ct$ and $1 \leq i \leq n$ such that $?a@i$ is at the i'th coordinate
  (the last component can handshake with the i'th component).

Obviously, ϕ' is of width 2 as required. Let us define $A' \overset{\text{def}}{=} (A'_1 \| \cdots \| A'_n \| A'_{n+1})_{\phi'}$.



init
$!a_{i_m}@i_m$
$z{\in}[0,0]$
$\phi(a_1,\ldots,a_n)$
$z{:=}0$
$reset(a_1,\ldots,a_n,i_m)$
$test(a_1,\ldots,a_n,i_1)$
$!a_{i_{m-1}}@i_{m-1}$
$z{\in}[0,0]$
$!a_{i_1}@i_1$
$reset(a_1,\ldots,a_n,i_{m-1})$
$test(a_1,\ldots,a_n,i_2)$
$!a_{i_{m-2}}@i_{m-2}$
$!a_{i_2}@i_2$
$reset(a_1,\ldots,a_n,i_1)$
$test(a_1,\ldots,a_n,i_m)$

Fig. 3. Automaton A_{n+1}

Theorem 4. *Reachability for CTA with an arbitrary synchronization function is reducible in polynomial time (and space) into reachability for CTA with synchronization function of width* 2*. The number of parallel components and of clocks is increased by one.*

Proof. Let $c = (s_1, \ldots, s_n, v)$ be a reachable configuration in A. A corresponding configuration $f(c)$ in A' is defined by $f(c) \overset{\text{def}}{=} (s_1, \ldots, s_n, init, v')$ where $v'(x) \overset{\text{def}}{=} v(x)$ for all $x \in C$ and $v'(z) \overset{\text{def}}{=} v(delay)$. Let $c^0 = (s^0_1, \ldots, s^0_n, v^0)$ be the initial configuration of A. We claim that a given configuration c is reachable (in A) from c^0 if and only if $f(c)$ is reachable (in A') from $f(c^0)$. The claim follows from the following observation.

Let $c = (s_1, \ldots, s_n, v)$ be an arbitrary configuration of A. It is easy to see that whenever $c \xrightarrow{a} c'$ for some $c' = (s'_1, \ldots, s'_n, v')$ by using a tuple $(a_1, \ldots, a_n)$ from $dom(\phi)$ then also $f(c) \xrightarrow{a} \circ (\xrightarrow{\tau})^{2m} f(c')$ where m is the number of active components in $(a_1, \ldots, a_n)$. In A' we can first perform the transition $init \overset{g',a,\{z\}}{\Longrightarrow} test(a_1, \ldots, a_n, i_1)$ where i_1 is the first active component. This transition is followed by a unique continuation according to the path in Figure 3, all guards (of the corresponding component automata) during the first part of the path are satisfied by our assumption that $c \xrightarrow{a} c'$ in A. The corresponding clocks are reset during the second part of the path, until $f(c')$ is finally reached. On the other hand, if $f(c)$ performs a sequence of moves starting with selecting

$$\begin{tikzcd}
\text{CTA} \arrow[r, shift left, "\text{blow up}"] \arrow[d, dash] & \text{1-TAPN*} \arrow[l, shift left, "\text{poly-time}"] \arrow[d, dash] \\
\text{mrCTA} \arrow[r, shift left, "\text{blow up}"] & \text{1-TAPN} \arrow[l, shift left, "\text{poly-time}"]
\end{tikzcd}$$

arbitrary ϕ: up to isomorphism

$$\begin{tikzcd}
\text{CTA} \arrow[r, shift left, "\text{poly-time}"] \arrow[d, dash] & \text{1-TAPN*} \arrow[l, shift left, "\text{poly-time}"] \arrow[d, dash] \\
\text{mrCTA} \arrow[r, shift left, "\text{poly-time}"] & \text{1-TAPN} \arrow[l, shift left, "\text{poly-time}"]
\end{tikzcd}$$

constant width ϕ: up to isomorphism
arbitrary ϕ: up to reachability

Fig. 4. Summary of results

a tuple $(a_1, \ldots, a_n)$ from $dom(\phi)$ such that we finally reach c'' where in the last component appears again the state *init*, we know that no time-elapsing steps were performed because of the clock z which is tested to zero by every transition. Moreover, for a selected tuple $(a_1, \ldots, a_n)$ this computation is unique and by similar arguments as above we know that there is also some configuration c' in A such that $c \overset{\phi(a_1,\ldots,a_n)}{\longrightarrow} c'$ and $f(c') = c''$. Similarly, time elapsing steps on both sides can be directly matched in the other automaton. As it can be seen from the construction, only one extra parallel component and one additional clock were introduced. $\square$

6 Conclusion

The main results of the paper are outlined in Figure 4. We have identified a naturally motivated subclass of concurrent (networks of) timed automata by introducing the mandatory reset feature and a natural superclass of 1-safe timed-arc Petri nets extended with testing arcs such that the corresponding classes coincide up to isomorphism of labelled transition systems. We have then studied more practically oriented questions of reachability and demonstrated polynomial time reductions between the corresponding classes in Figure 4.

Our study justifies that it is interesting to investigate the extension of timed-arc Petri nets with testing arcs. This feature is present in the untimed Petri net model (using a standard trick) but it is missing when time features are added. We claim that all the classes CTA, mrCTA, 1-TAPN and 1-TAPN* are polynomially equivalent w.r.t. reachability. The answer to the question of polynomial equivalence w.r.t. reachability for (unbounded) TAPN and TAPN* seems to be also positive, even though more involved techniques need to be used to establish the reduction. In the future work we plan to consider TAPN extended with urgency and see how they compare to networks of timed automata with invariants. We will also investigate which TCTL properties are preserved by the polynomial time reductions presented in this paper and what is the relationship between TAPN and Merlin's time Petri nets.

Finally, let us draw our attention to an interesting observation. Already for concurrent finite automata (untimed), the reachability problem is PSPACE-complete [18] and adding time features (CTA) does not increase the theoretical complexity of the problem [2]. The results of this paper enable to draw a similar conclusion also for the case of 1-safe Petri nets: the reachability problem for untimed 1-safe Petri nets (a formalism which can model concurrency) is PSPACE-complete [15] and adding time features does not increase its complexity.

Acknowledgments. I would like to thank Luca Aceto and Ole H. Jensen for reading a preliminary draft of this paper, and the anonymous referees for their comments and suggestions.

References

[1] P.A. Abdulla and A. Nyln. Timed Petri nets and BQOs. In *Proc. of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *LNCS*, pages 53–70. Springer-Verlag, 2001.

[2] L. Aceto and F. Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Programming*, 52–53:7–51, 2002.

[3] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[4] A. Arnold. *Finite Transition Systems.* Prentice-Hall, 1994.

[5] B. Berard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.

[6] 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*, 2004. To appear.

[7] T. Bolognesi, F. Lucidi, and S. Trigila. From timed Petri nets to timed LOTOS. In *Proc. of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification (Ottawa 1990)*, pages 1–14. 1990.

[8] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *International Symposium: Compositionality - The Significant Difference, Malente (Holstein, Germany)*, volume 1536 of *LNCS*, 1998.

[9] P. Bouyer. Untameable timed automata! In *Proc. of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03)*, volume 2607 of *LNCS*, pages 620–631. Springer-Verlag, 2003.

[10] Fred D.J. Bowden. Modelling time in Petri nets. In *Proc. of the Second Australia-Japan Workshop on Stochastic Models*, 1996. `http://www.itr.unisa.edu.au/∼fbowden/pprs/stomod96/`.

[11] M. Boyer and M. Diaz. Non equivalence between time Petri nets and time stream Petri nets. In *Proc. of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99)*, pages 198–207. IEEE Computer Society, 1999.

[12] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *Proc. of the 10th International Conference on Computer-Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 546–550. Springer–Verlag, 1998.

[13] N. Busi and G.M. Pinna. Process semantics for place/transition nets with inhibitor and read arcs. *Fundamenta Informaticae*, 40(2–3):165–197, 1999.

[14] F. Cassez and O.H. Roux. Structural translation of time petri nets into timed automata. In *Workshop on Automated Verification of Critical Systems (AVoCS'04)*, ENTCS. Elsevier, 2004.

[15] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1-2):117–136, 1995.

[16] S. Haar, F. Simonot-Lion, L. Kaiser, and J. Toussaint. Equivalence of timed state machines and safe time Petri nets. In *Proc. of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 119–126, 2002.

[17] H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *Proc. of the 14th International Conference on Application and Theory of Petri Nets (ICATPN'93)*, volume 691 of *LNCS*, pages 282–299, 1993.

[18] D. Kozen. Lower bounds for natural proof systems. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science*, pages 254–266. IEEE, 1977.

[19] F. Laroussinie and K.G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In *Proc. of the FIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII)*, pages 439–456. Kluwer, B.V., 1998.

[20] F. Laroussinie, N. Markey, and Ph. Schnoebelen. On model checking durational Kripke structures. In *Proc. of the 5th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'02)*, volume 2303 of *LNCS*, pages 264–279. Springer-Verlag, 2002.

[21] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[22] D. Lime and O.H. Roux. State class timed automaton of a time Petri net. In *Proc. of the 10th International Workshop on Petri Net and Performance Models (PNPM'03)*, pages 124–133, 2003.

[23] M. Nielsen, V. Sassone, and J. Srba. Properties of distributed timed-arc Petri nets. In *Proc. of the 21st International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01)*, volume 2245 of *LNCS*, pages 280–291. Springer-Verlag, 2001.

[24] O. Roux, D. Lime, and G. Gardey. Romeo: a software studio for time Petri net analysis. `http://www.irccyn.ec-nantes.fr/irccyn/d/en/equipes/TempsReel/logs/software-2-romeo`.

[25] V. Valero Ruiz, D. de Frutos Escrig, and O. Marroquin Alonso. Decidability of properties of timed-arc Petri nets. In *Proc. of the 21st International Conference on Application and Theory of Petri Nets (ICATPN'00)*, volume 1825 of *LNCS*, pages 187–206. Springer-Verlag, 2000.

[26] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proc. of the 8th International Workshop on Petri Net and Performance Models (PNPM'99)*, pages 188–196, 1999.

[27] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Proc. of the 13th Annual Symposim on Theoretical Aspects of Computer Science (STACS'96)*, volume 1046 of *LNCS*, pages 347–359. Springer-Verlag, 1996.

[28] W. Vogler. Partial order semantics and read arcs. *Theoretical Computer Science*, 286(1):33–63, 2002.

[29] J. Wang. *Timed Petri Nets, Theory and Application.* Kluwer Academic Publishers, 1998.