FI MU

Faculty of Informatics
Masaryk University Brno

An Efficient Translation of Timed-Arc Petri Nets to Networks of Timed Automata

by

Joakim Byg
Kenneth Yrke Jørgensen
Jiří Srba

FI MU Report Series FIMU-RS-2009-06

October 2009

Publications in the FI MU Report Series are in general accessible via WWW:

http://www.fi.muni.cz/reports/

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

An Efficient Translation of Timed-Arc Petri Nets to Networks of Timed Automata

Joakim Byg
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
joakim.byg@gmail.com

Kenneth Yrke Jørgensen
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
kyrke@cs.aau.dk

Jiří Srba
Aalborg University
Selma Lagerlöfs Vej 300
9220 Aalborg East
srba@cs.aau.dk

October 31, 2009

Abstract

Bounded timed-arc Petri nets with read-arcs were recently proven equivalent to networks of timed automata, though the Petri net model cannot express urgent behaviour and the described mutual translations are rather inefficient. We propose an extension of timed-arc Petri nets with invariants to enforce urgency and with transport arcs to generalise the read-arcs. We also describe a novel translation from the extended timed-arc Petri net model to networks of timed automata. The translation is implemented in the tool TAPAAL and it uses UPPAAL as the verification engine. Our experiments confirm the efficiency of the translation and in some cases the translated models verify significantly faster than the native UPPAAL models do.

1 Introduction

Time dependent models have been intensively studied because of the current needs in software verification and development of embedded applications where several reliability and safety requirements depend, to a large extent, on the timing aspects. Among the most studied time dependent models are timed automata [3] and different time extensions of Petri nets (see e.g. [14]). A recent overview comparing these models has been given in [20].

We consider a particular extension of the Petri net model called *Timed-Arc Petri Nets* (TAPN) [7, 11] where an age (a real number) is assigned to each token in the net and time intervals on arcs restrict the ages of tokens that can be used to fire a transition. Recent studies show that bounded TAPN (where the maximum number of tokens in the net is a priori given) offer a similar expressive power as networks of timed automata, even though the models are conceptually different and suitable for modelling of different systems. Sifakis and Yovine [18] provided a translation of 1-safe timed-arc Petri nets into timed automata which preserves strong timed bisimilarity but their translation causes an exponential blow up in the size. Srba established in [19] a strong relationship (up to isomorphism of timed transition systems) between networks of timed automata and a superclass of 1-safe TAPN extended with read-arcs. For reachability questions the reductions work in polynomial time. Recently Bouyer et al. [8] presented a reduction from bounded TAPN (with read-arcs) to 1-safe TAPN (with read-arcs), which preserves timed language equivalence (over finite words, infinite words and non-Zeno infinite words). Nevertheless the translations described in these papers are inefficient from a practical point of view as they either cause an exponential blow-up in the size or create a new parallel component with a fresh local clock (or more if the net is not 1-safe) for *each place* in the net, a situation where even most developed tools like UPPAAL [21] show often a poor performance. One limitation of the TAPN model is the impossibility to express urgent behaviour (a TAPN model can always in any marking delay for ever without taking any discrete transitions). While on one side this makes some problems like coverability and boundedness decidable even for unbounded nets [15, 2, 1, 8], it considerably limits the modelling power.

In this paper we extend the TAPN model with two new features: *invariants*[1] on places to enforce urgent behaviour and *transport arcs* that generalise the previously studied read-arcs [19, 8]. We then suggest a novel translation of TAPN to networks of timed automata where a fresh parallel component (with a local clock) is created for *every token* in the net. This is a conceptually orthogonal approach to the ones discussed in the previ-

[1] Invariants in our setting are time bounds restricting the ages of tokens in certain places. They should not be confused with transition/place invariant techniques studied in the theory of (untimed) Petri nets.

ous works and it relies on different reduction techniques. The proposed translation also transforms safety and liveness logical formulae into equivalent formulae on networks of timed automata. One of the main advantages of this approach is the ability to use the *active clock reduction* and the *symmetry reduction* techniques available in the rich theory of timed automata.

The theory described in this paper translates TAPN models to UPPAAL-style of timed automata with handshake synchronization because UPPAAL is probably the most frequently used industrial-strength tool for verification of timed automata. For this reason, we chose at the moment not to use tools offering more general notions of synchronization like e.g. KRONOS [9] and our experiments confirm that the translation to timed automata with handshake synchronization was indeed a good choice as the verification using this approach is rather efficient. The suggested translations were implemented in a new tool TAPAAL [10], freely available at `www.tapaal.net`, which offers modelling, simulation and verification of timed-arc Petri nets with continuous time. We report here on two experiments: verification of the Fischer's mutual exclusion algorithm and the alternating bit protocol. The results are promising and the translated timed automata models verify in fact considerably faster than the native UPPAAL models do.

Related Tools. There is one related tool prototype for verification of timed-arc Petri nets mentioned in [2] where the authors discuss a coverability algorithm for general (unbounded) nets, though without any urgent behaviour. The tool does not seem to be maintained anymore. Time features (time stamps) connected to tokens can be modelled also in Coloured Petri Nets using CPN Tools [12], however, only discrete time semantics is implemented in CPN Tools with a limited support for the automatic analysis.

2 Basic Definitions

A *timed labelled transition system* (TLTS) is a triple $T = (S, \mathcal{A}ct, \longrightarrow)$ where S is a set of *states*, $\mathcal{A}ct$ is a set of *actions* where $\mathcal{A}ct \cap \mathbb{R}^{\geq 0} = \emptyset$ and $\mathbb{R}^{\geq 0}$ are nonnegative real numbers, and $\longrightarrow \subseteq S \times (\mathcal{A}ct \cup \mathbb{R}^{\geq 0}) \times S$ is a *transition relation*.

We let a, $a_0, a_1, \ldots$ range over $\mathcal{A}ct$ and $d, d_0, d_1, \ldots$ over $\mathbb{R}^{\geq 0}$. We write $s \xrightarrow{a} s'$ if $(s, a, s') \in \longrightarrow$ for the *discrete transitions* and $s \xrightarrow{d} s'$ if $(s, d, s') \in \longrightarrow$ for the *delay transitions*. We use the notations $s \xrightarrow{a}$ and $s \xrightarrow{d}$ if there exists some state s' such that $s \xrightarrow{a} s'$ and $s \xrightarrow{d} s'$, respectively. By $s \longrightarrow s'$ we mean that either $s \xrightarrow{a} s'$ for some

$a \in \mathcal{A}ct$ or $s \xrightarrow{d} s'$ for some delay d. Let $s \in S$ and $d \in \mathbb{R}^{\geq 0}$. By $s[d]$ we denote the unique (here we impose the standard *time-determinism* assumption—see e.g. [5]) state s' such that $s \xrightarrow{d} s'$, provided that the delay d is possible from s.

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

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

The set $\mathcal{I}_{\mathrm{Inv}}$ of *invariants* is a subset of intervals that include 0.

2.1 Logics for Safety and Liveness Properties

We shall now define a subset of Computation Tree Logic (CTL) used in the tool TAPAAL [10] (essentially mimicking the logic used in UPPAAL, except for the *leads-to* operator). Let $\mathcal{AP}$ be the set of atomic propositions. The logical formulae are given by the following abstract syntax

$$\begin{aligned} \psi &::= \ \mathsf{EF}\,\varphi \mid \mathsf{EG}\,\varphi \mid \mathsf{AF}\,\varphi \mid \mathsf{AG}\,\varphi \\ \varphi &::= \ p \mid \neg\varphi \mid \varphi \wedge \varphi \end{aligned}$$

where $p \in \mathcal{AP}$ and EF, EG, AF and AG are the standard CTL temporal operators.

The semantics is defined with respect to a given TLTS $T = (S, \mathcal{A}ct, \longrightarrow)$ together with a labelling function $\mu : S \rightarrow 2^{\mathcal{AP}}$ which assigns a set of true atomic propositions to each state. The satisfaction relation $s \models \psi$ for a state $s \in S$ and a formula ψ is defined inductively as follows:

- $s \models p$ iff $p \in \mu(s)$,
- $s \models \neg\varphi$ iff $s \not\models \varphi$,
- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$,
- $s \models \mathsf{EF}\,\varphi$ iff $s \longrightarrow^* s'$ and $s' \models \varphi$
- $s \models \mathsf{EG}\,\varphi$ iff there is a (finite or infinite) alternating run ρ of the form

  $$s = s_1 \xrightarrow{d_1} s_1' \xrightarrow{a_1} s_2 \xrightarrow{d_2} s_2' \xrightarrow{a_2} s_3 \xrightarrow{d_3} s_3' \xrightarrow{a_3} s_4 \xrightarrow{d_4} s_4' \xrightarrow{a_4} \ldots$$

  such that for all i and for all d, $0 \leq d \leq d_i$, we have $s_i[d] \models \varphi$ and

  (i) ρ is infinite, or

(ii) ρ is finite and ends in s_k where for all $d \in \mathbb{R}^{\geq 0}$ we have $s_k \xrightarrow{d}$ and $s_k[d] \models \varphi$, or

(iii) ρ is finite and ends in a state s' (where s' is either of the form s_k or s'_k) such that whenever $s' \xrightarrow{d} s'[d]$ is possible for a $d \in \mathbb{R}^{\geq 0}$ then $s'[d] \models \varphi$ and there is no state s'' such that $s'[d] \xrightarrow{a} s''$ for any $a \in \mathcal{A}ct$,

- $s \models \mathsf{AF}\,\varphi$ iff $s \not\models \mathsf{EG}\,\neg\varphi$, and
- $s \models \mathsf{AG}\,\varphi$ iff $s \not\models \mathsf{EF}\,\neg\varphi$.

Remark 2.1. *The formula* $\mathsf{EG}\,\varphi$ *means that there exists a* maximal *run such that at any point the formula* φ *is satisfied. The conditions (i), (ii) and (iii) list the three possibilities for a run to be maximal: (i) it consists of an infinite alternating sequence of actions and time delays, or (ii) it ends in a state where time can diverge, or (iii) it ends in a state from which no discrete transitions are possible after any time delay (this includes time-locks).*

2.2 Timed-Arc Petri Nets

A *Timed-Arc Petri Net with transport arcs and place invariants* (TAPN) is a tuple $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$, 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*, $c : F|_{P\times T} \longrightarrow \mathcal{I}$ is a function assigning a time interval to every arc from a place to a transition, $F_{tarc} \subseteq (P \times T \times P)$ is the set of *transport arcs* that satisfy for all $(p, t, p') \in F_{tarc}$ and all $r \in P$:

$$\big((p,t,r) \in F_{tarc} \Rightarrow p' = r\big) \land \big((r,t,p') \in F_{tarc} \Rightarrow p = r\big) \land (p,t) \notin F \land (t,p') \notin F$$

$c_{tarc} : F_{tarc} \longrightarrow \mathcal{I}$ is a function assigning a time interval to every transport arc, and $\iota : P \longrightarrow \mathcal{I}_{Inv}$ is an *invariant assignment* of invariants to places.

Remark 2.2. *The conditions imposed on the transport arcs guarantee for any given* p *and* t *that if there is a transport arc of the form* (p, t, p') *or* (p'', t, p) *then the places* p' *and* p'' *are unique. Whenever the places* p' *or* p'' *are not relevant for the context, we shall simply denote the transport arcs as* $(p, t, \_)$ *or* $(\_, t, p)$.

The *preset* of a transition t in the net is defined as ${}^\bullet t = \{p \in P \mid (p,t) \in F \lor (p,t,\_) \in F_{tarc}\}$, and the *postset* of a transition t is defined as $t^\bullet = \{p \in P \mid (t,p) \in F \lor (\_,t,p) \in F_{tarc}\}$. Without loss of generality assume that $|{}^\bullet t \cup t^\bullet| > 0$ for any $t \in T$. By $\mathcal{B}(\mathbb{R}^{\geq 0})$ we denote the set of finite multisets on $\mathbb{R}^{\geq 0}$. For $B \in \mathcal{B}(\mathbb{R}^{\geq 0})$ and $d \in \mathbb{R}^{\geq 0}$ we let $B + d \overset{\text{def}}{=} \{b + d \mid b \in B\}$.

Let $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$ be a TAPN. A *marking* M on the net N is a function $M \colon P \to \mathcal{B}(\mathbb{R}^{\geq 0})$ such that every $p \in P$ and every $x \in M(p)$ satisfy $x \in \iota(p)$. Each place is thus assigned a certain number of tokens, and each token is annotated with a real number (*age*). We moreover consider only markings such that all their tokens satisfy the place invariants imposed by the invariant assignment ι. By $|M|$ we denote the total number of tokens in the marking M. The set of all markings on N is denoted by $\mathcal{M}(N)$. For a finite marking M (where $|M| < \infty$) we also use an alternative multiset notation $M = \{(p_1, r_1), (p_2, r_2), \ldots, (p_k, r_k)\}$ where $p_i \in P$ and $r_i \in \mathbb{R}^{\geq 0}$, which lists explicitly all tokens in the net by naming their positions and ages. A *marked TAPN* is a pair (N, M_0) where N is TAPN and M_0 is an initial marking. As *initial markings* we allow only markings with tokens of age 0.

Let us now outline the dynamics of TAPNs. We introduce two types of transition rules: *firing* of a transition and *time delay*.

For a TAPN N we say that a transition $t \in T$ is *enabled* in a marking M if

- in all places $p \in {}^\bullet t$ there is a token x such that its age belongs to the time interval on the arc from p to t, and
- if there is a transport arc of the form (p, t, p') then moreover the age of the token in p satisfies the invariant imposed by p'.

If a transition t is enabled then it can *fire*. It consumes one token (of an appropriate age) from each place in ${}^\bullet t$, and produces one new token to every place in $t^\bullet$. The age of the newly produced token is either 0 for the standard arcs, or it preserves the age of the consumed token for transport arcs.

Another behaviour of the net is a so-called *time delay* where all tokens in the net grow simultaneously older by a given time factor (a real number in general). A time delay is allowed only as long as invariants in all places are satisfied.

Example 2.3. *Consider the marked TAPN from Fig. 1. There are 8 places (drawn as circles) and 6 transitions (drawn as rectangles) that are connected either by standard arcs (such that every arc from a place to a transition contains a time interval) or transport arcs like the one from* p_1 *to* p_3 *via* t_1*. Transport arcs via a given transition are numbered (the symbol* $:1$ *after the interval on the arc from* p_1 *to* t_1 *and the symbol* 1 *on the arc from* t_1 *to* p_3*) so that the routes for tokens that do not change their age after transition firing are clearly identified. The initial marking contains only one token in place* p_0 *of age* 0 *time units. Clearly, before* t_0 *can fire the net has to delay between* 1 *to* 3 *time units and after its firing two new tokens of age* 0 *are produced into*

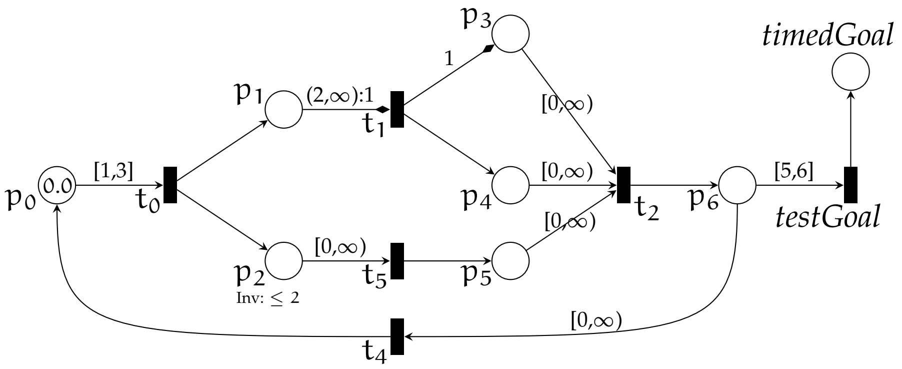


Figure 1: Example of a marked TAPN

p_1 *and* p_2. *In fact, a longer delay of say* 5 *time units is also possible but then the transition* t_0 *will not be enabled again and the token in* p_0 *is sometimes referred to as a* dead token. *The place* p_2 *contains an invariant ensuring that tokens in that place cannot grow older than* 2 *time units. The other places do not show any invariant information, which implicitly means that their associated invariant is* $[0, \infty)$. *The transport arc between* p_1 *and* p_3 *ensures that when* t_1 *is fired the age of the token produced into* p_3 *is equal to the age of the token consumed in* p_1 *(the token produced to* p_4 *is of age* 0*).*

Transition Firing. In a marking M, we can fire a transition t if it is enabled, i.e.

$$\forall p \in {}^\bullet t.\ \exists x \in M(p).\ [x \in c(p,t) \lor (x \in c_{tarc}(p,t,p') \land x \in \iota(p'))] \ .$$

Before firing t, we fix the sets $C_t^-(p)$ and $C_t^+(p)$ for all places $p \in P$ so that they satisfy the following equations (note that all operations are on multisets, and there may be several options for fixing these sets):

- for every $p \in P$ such that $(p,t) \in F$
  $C_t^-(p) = \{x\}$ where $x \in M(p)$ and $x \in c(p,t)$,
- for every $p \in P$ such that $(t,p) \in F$
  $C_t^+(p) = \{0\}$, and
- for every $p, p' \in P$ such that $(p,t,p') \in F_{tarc}$
  $C_t^-(p) = \{x\} = C_t^+(p')$ where $x \in M(p)$, $x \in c_{tarc}(p,t,p')$ and $x \in \iota(p')$;
- in all other cases (when the place in the argument is unrelated to the firing of the transition t) we set the above sets to $\emptyset$.

Firing a transition t in the marking M yields a new marking M' defined as

$$\forall p \in P.\ M'(p) \overset{\text{def}}{=} \Big(M(p) \setminus C_t^-(p)\Big) \cup C_t^+(p) \ .$$

Time Delays. In a marking M we can let time pass by $d \in \mathbb{R}^{\geq 0}$ time units if

$$\forall p \in P.\ \forall x \in M(p).\ (x + d) \in \iota(p)$$

and this time delay then yields a marking M' defined as

$$\forall p \in P.\ M'(p) \overset{\text{def}}{=} M(p) + d\ .$$

A given TAPN $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$ generates a TLTS $T(N) \overset{\text{def}}{=} (\mathcal{M}(N), T, \longrightarrow)$ where states are markings on N, the set of actions is T, and the transition relation $\longrightarrow$ is defined by $M \xrightarrow{t} M'$ whenever the firing of a transition t in a marking M yields a marking M', and $M \xrightarrow{d} M'$ whenever a time delay of d time units in a marking M yields a marking M'.

In a marked TAPN (N, M_0) we say that a marking M is reachable iff $M_0 \longrightarrow^* M$. The set of all reachable markings from marked TAPN (N, M_0) is denoted $\mathcal{M}(N, M_0)$. A marked net N is k-*bounded* if the total number of tokens in any of its reachable markings is less or equal to k. A marked net is called *bounded* if it is k-bounded for some k. A net N is of *degree* k if every transition $t \in T$ has exactly k incoming and exactly k outgoing arcs, formally $|{}^\bullet t| = |t^\bullet| = k$.

In order to argue about the validity of logical formulae on transition systems generated by timed-arc Petri nets, we also have to define the set of atomic propositions $\mathcal{AP}$ and the labelling function $\mu : \mathcal{M}(N) \to 2^{\mathcal{AP}}$. We let $\mathcal{AP} \overset{\text{def}}{=} \{p \bowtie n \mid p \in P, n \in \mathbb{N} \text{ and } \bowtie \in \{<, \leq, =, \geq, >\}\}$. The interpretation is that a proposition $(p \bowtie n)$ is true in marking M iff the number of tokens in the place p satisfies the proposition in question with respect to n, formally $\mu(M) \overset{\text{def}}{=} \{(p \bowtie n) \mid |M(p)| \bowtie n\}$, where $\bowtie$ is one of the (standard mathematical) operators in the above definition.

Given a marked TAPN (N, M_0) and a formula ψ, we shall write $M_0 \models_N \psi$ (or $M_0 \models \psi$ if N is clear from the context) whenever the marking M_0 satisfies the formula ψ in the TLTS $T(N)$.

Consider again the marked TAPN from Fig. 1. It is easy to verify that it satisfies e.g. the formula $\mathsf{EF}\ (p_6 = 1)$ as the place p_6 can be easily marked. In our logic we do not consider queries that involve any timing information of tokens but such formulae can be still verified with the presented logic by adding new testing transitions like the one called *testGoal* moving tokens of the specified age from the place p_6 to *timedGoal*. Now the property whether p_6 can become marked with a token of age between 5 and 6 time units can be expressed as the formula $\mathsf{EF}\ (\textit{timedGoal} = 1)$. Similarly, by introducing a new place with a token and resetting its age when a certain transition is fired, one can measure the duration before some other transition is fired.

Remark 2.4. *In standard P/T Petri nets there is a 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 with only standard arcs as consuming a (timed) token and returning it back resets its age. Hence an extension of the model with* read-arcs *was suggested in [19, 8]. A read-arc in TAPN setting is a special arc from a place to a transition which is labelled by a time interval. The semantics is that the transition can fire only if a token with its age in the given interval is present in the input place of the read-arc, however, the token is not consumed nor reset when the transition is fired. It is shown in [19, 8] that timed automata and bounded TAPN with read-arcs are equally expressive. Transport arcs, newly introduced in this paper, generalize the notion of read-arcs because a read-arc can be simulated by a pair of transport arcs which consume a token and return it back without resetting its age (the same trick as in P/T nets). Transport arcs do not add any expressive power as we show in this paper that bounded TAPN with transport arcs can be also translated to timed automata. On the other hand, transport arcs are convenient for the modelling purposes because the encoding tricks used in simulating transport arcs by read-arcs are complex and they double the number of tokens in the net (as one token is used to simulate the token position and the other one to remember its age).*

2.3 Networks of Timed Automata

Let C be a finite set of *clocks*. A *(time) valuation* of clocks from C is a function $v : C \to \mathbb{R}^{\geq 0}$. Let v be a valuation and $d \in \mathbb{R}^{\geq 0}$. We define a valuation $v + d : C \to \mathbb{R}^{\geq 0}$ by $(v + d)(x) \overset{\text{def}}{=} v(x) + d$ for every $x \in C$. For every set $R \subseteq C$ we define a valuation $v[R := 0] : C \to \mathbb{R}^{\geq 0}$ by $v[R := 0](x) \overset{\text{def}}{=} v(x)$ for $x \in C \smallsetminus R$ and $v[R := 0](x) \overset{\text{def}}{=} 0$ for $x \in R$.

A *clock guard* is a partial function $g : C \hookrightarrow \mathcal{I}$ assigning a time interval to selected clocks. We denote the set of all clock guards as $\mathcal{G}(C)$. An *invariant* is a clock guard g where for every $x \in C$ holds $g(x) \in \mathcal{I}_{\mathrm{Inv}}$ whenever $g(x)$ is defined. The set of all invariants is denoted by $\mathcal{G}_{\mathrm{Inv}}(C)$. We say that a valuation v satisfies a guard $g \in \mathcal{G}(C)$ (written $v \models g$) iff $v(x) \in g(x)$ for all $x \in \mathrm{dom}(g)$. To specify a guard g that only constrains the values of one clock x, we often use the notation $x \in I$ where $I = g(x)$.

A *timed automaton* (TA) is a tuple $A = (L, \mathcal{A}ct, C, \longrightarrow, \iota, \ell^0)$ where L is a finite set of *locations*, $\mathcal{A}ct$ is a finite set *actions* such that $L \cap \mathcal{A}ct = \emptyset$, C is a finite set of *clocks*, $\longrightarrow \subseteq L \times \mathcal{G}(C) \times \mathcal{A}ct \times 2^C \times L$ is a finite *transition relation* written $\ell \xrightarrow{g,a,R} \ell'$ for $(\ell, g, a, R, \ell') \in \longrightarrow$, $\iota : L \to \mathcal{G}_{Inv}(C)$ is an *invariant assignment* of clock guards to the locations, and $\ell^0 \in L$ is an *initial* location.

A *configuration* of a timed automaton A is a pair (ℓ, ν) where $\ell \in L$ is a location and $\nu : C \to \mathbb{R}^{\geq 0}$ is a clock valuation on C such that the location ℓ satisfies the respective invariant, i.e., $\nu \models \iota(\ell)$. We denote the set of all configurations of A by *Conf*(A). An *initial configuration* of A is (ℓ^0, ν^0) such that $\nu^0(x) \overset{\text{def}}{=} 0$ for all $x \in C$. We assume that the initial configuration always satisfies the invariant of the location ℓ^0, i.e., $(\ell^0, \nu^0) \in \mathit{Conf}(A)$.

A timed automaton $A = (L, \mathcal{A}ct, C, \longrightarrow, \iota, \ell^0)$ determines a TLTS $T(A) \overset{\text{def}}{=} (\mathit{Conf}(A), \mathcal{A}ct, \longrightarrow)$ where states are configuration of A and the transition relation $\longrightarrow$ is defined by

$$\begin{array}{ll} (\ell, \nu) \xrightarrow{a} (\ell', \nu[R := 0]) & \text{if } \ell \xrightarrow{g,a,R} \ell' \text{ in } A \text{ s.t. } \nu \models g \text{ and } \nu[R := 0] \models \iota(\ell') \\ (\ell, \nu) \xrightarrow{d} (\ell, \nu + d) & \text{if } d \in \mathbb{R}^{\geq 0} \text{ and for all } d' \in [0, d] \text{ we have } \nu + d' \models \iota(\ell). \end{array}$$

We shall adopt the handshake communication scheme as it is used in the tool UPPAAL [21] for defining a parallel composition of automata. In the semantics, we consider only synchronization moves as independent moves of single components are not necessary for the reduction.

Let $A_1, \ldots, A_n$ be timed automata where (for all i, $1 \leq i \leq n$) $A_i = (L_i, \mathcal{A}ct, C, \longrightarrow_i, \iota_i, \ell_i^0)$ and where $\mathcal{A}ct$ and C are fixed sets of actions and clocks, respectively. We moreover require that $\mathcal{A}ct$ is of the form $\mathcal{A}ct = \mathcal{A}ct_! \cup \mathcal{A}ct_?$ where $\mathcal{A}ct_! \overset{\text{def}}{=} \{a! \mid a \in \mathit{Chan}\}$ and $\mathcal{A}ct_? \overset{\text{def}}{=} \{a? \mid a \in \mathit{Chan}\}$ for a given nonempty set of *channel names Chan*. A *network of timed automata* (NTA) is a parallel composition of $A_1, \ldots, A_n$ denoted by $P = A_1 | \cdots | A_n$. Note that it is allowed to share the names of locations in different parallel components.

A *configuration* is a tuple $(\ell_1, \ldots, \ell_n, \nu)$ where $\ell_i \in L_i$ for all $1 \leq i \leq n$ and $\nu : C \to \mathbb{R}^{\geq 0}$ is a clock valuation on C such that for every i, $1 \leq i \leq n$, we have $\nu \models \iota_i(\ell_i)$. We denote the set of all configurations of P by *Conf*(P). An *initial configuration* of P is $(\ell_1^0, \ldots, \ell_n^0, \nu^0)$ such that $\nu^0(x) \overset{\text{def}}{=} 0$ for all $x \in C$. As before we assume that $(\ell_1^0, \ldots, \ell_n^0, \nu^0) \in \mathit{Conf}(P)$.

An NTA P determines a TLTS $T(P) \overset{\text{def}}{=} (\mathit{Conf}(P), \mathit{Chan}, \longrightarrow)$ where states are the configurations of P, the discrete transitions are labelled by channel names, and the transition relation $\longrightarrow$ is defined by

- $(s_1, \ldots, s_j, \ldots, s_k, \ldots s_n, \nu) \xrightarrow{a} (s_1, \ldots, s_j', \ldots, s_k', \ldots, s_n, \nu')$
  for $1 \leq j \neq k \leq n$ whenever

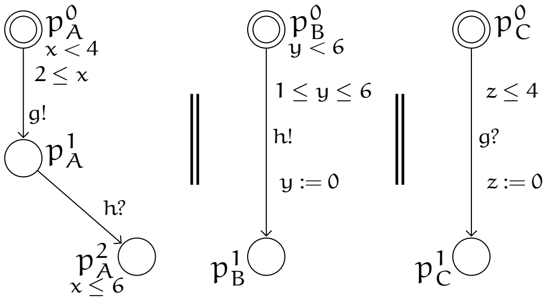


Figure 2: Example of an NTA

- $s_j \xrightarrow{g_j, a!, R_j}_j s'_j$ and $v \models g_j$,
- $s_k \xrightarrow{g_k, a?, R_k}_k s'_k$ and $v \models g_k$,
- $v' = v[R_j \cup R_k := 0]$, and $(s_1, \ldots, s'_j, \ldots, s'_k, \ldots, s_n, v') \in \mathit{Conf}(P)$

- $(s_1, \ldots, s_n, v) \xrightarrow{d} (s_1, \ldots, s_n, v + d)$
  if $d \in \mathbb{R}^{\geq 0}$ and $(s_i, v) \xrightarrow{d}_i (s_i, v + d)$ for all i, $1 \leq i \leq n$.

Example 2.5. *Consider the NTA in Fig. 2 with three parallel components* A, B *and* C. *We draw the parallel components as graphs where nodes represent locations together with their invariants and edges decorated by guards, synchronisation channels and clock updates represent the transition relation. The initial location of each component is marked with a double circle. In the following example of a computation in the network*

$$(p_A^0, p_B^0, p_C^0, [x = 0, y = 0, z = 0]) \xrightarrow{3} (p_A^0, p_B^0, p_C^0, [x = 3, y = 3, z = 3]) \xrightarrow{g}$$

$$(p_A^1, p_B^0, p_C^1, [x = 3, y = 3, z = 0]) \xrightarrow{2.4} (p_A^1, p_B^0, p_C^1, [x = 5.4, y = 5.4, z = 2.4]) \xrightarrow{h}$$

$$(p_A^2, p_B^1, p_C^1, [x = 5.4, y = 0, z = 2.4]) \xrightarrow{0.6} (p_A^2, p_B^1, p_C^1, [x = 6, y = 0.6, z = 3])$$

we notice that in the last configuration the network is stuck as no further synchronization is possible and because of the invariant $x \leq 6$ *in place* p_A^2 *time cannot delay either.*

In order to argue about validity of logical formulae on transition systems generated by networks of timed automata P, we have to define the set of atomic proposition $\mathcal{AP}$ and the labelling function $\mu : \mathit{Conf}(P) \to 2^{\mathcal{AP}}$. We let $\mathcal{AP} \stackrel{\text{def}}{=} \{(\#\ell \bowtie n) \mid \ell \in \cup_{i=1}^{n} L_i,\ n \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}\}$. The interpretation is that a proposition $(\#\ell \bowtie n)$ is true in a given configuration iff the number of parallel components that are currently in the location ℓ respects the given proposition with respect to n.

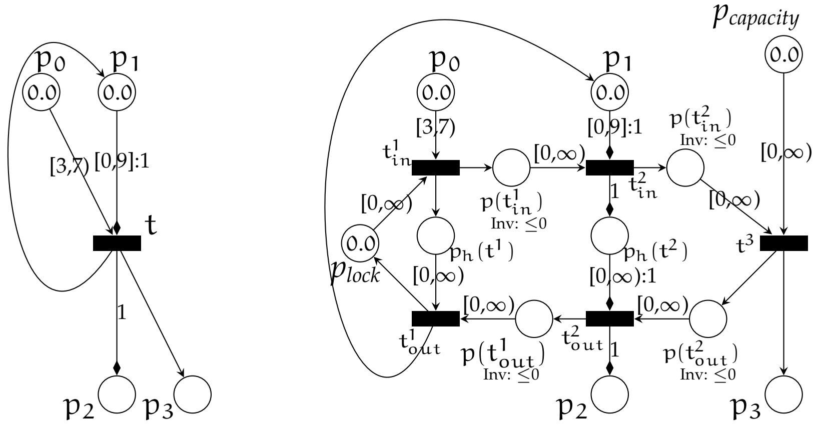


Figure 3: An example of a 3-bounded net and the corresponding net of degree 2

3 From Bounded TAPN to NTA

In this section we shall describe a reduction from bounded timed-arc Petri nets with invariants and transport arcs to networks of timed automata. We first describe a reduction from bounded nets to nets where each transition has exactly two input and two output places. In the second step this reduction is followed by a reduction to networks of timed automata.

3.1 From k-bounded TAPN to TAPN of Degree 2

To translate a given k-bounded TAPN with transitions that have more than two input or output places into a TAPN of degree 2 we have to simulate a single transition firing in the original net by a series of transitions in the net of degree 2. The problem is that when firing a given transition in a number of steps, other transition firings may interleave—thus some extra behaviour can be introduced. To prevent this from happening, we introduce a new mutex-like place called p_{lock}, which contains a token that is consumed before the sequence of transition firings begins and the token is returned back after the simulation of the selected transition is ended.

The translation is demonstrated in Fig. 3 where a simple 3-bounded TAPN is translated into a TAPN of degree 2. The idea is that the token in the place p_{lock} will travel through intermediate places $p(t^1_{in})$, $p(t^2_{in})$, $p(t^2_{out})$, $p(t^1_{out})$ and finally return to p_{lock}. When the first transition $p(t^1_{in})$ is fired, a token of a suitable age from p_0 is consumed and placed in the holding place $p_h(t^1)$, then a token from p_1 is consumed and placed in $p_h(t^2)$. Because $|^\bullet t| < |t^\bullet|$ a special place called $p_{capacity}$ (used as a repository of the presently unused tokens) is created. By firing the transition t^3 a new token of age 0 is produced in p_3. And finally the tokens placed in $p_h(t^2)$ and $p_h(t^1)$ are moved to the appropriate output places by firing the transitions t^2_{out} and t^1_{out}. Note that because of the invariants on the intermediate places, time cannot elapse during such a series of transition firing and so the age of the token in p_1 that is moved via the two transport arcs into p_2 is preserved. Notice that the use of holding places is essential as without them a token from p_0 can be consumed by t^1_{in} while creating a new token in p_1. This may allow firing of t^2_{in} even if there were no tokens in p_1 in the original net. Also notice that by this construction we may introduce extra deadlocks (e.g. if there is no token present in p_1 and the transition t^1_{in} is fired). Nevertheless, for the verification of safety properties we can detect such situations as demonstrated in what follows.

Let us introduce some notation for a transition $t \in T$. We fix a set

$$\begin{aligned}
\mathit{Pairing}(t) =& \{(p, I, p', \mathit{tarc}) \mid (p, t, p') \in F_{tarc}, I = c_{tarc}(p, t, p')\} \cup \\
& \{(p_1, I_1, p'_1, \mathit{normal}), \ldots, (p_m, I_m, p'_m, \mathit{normal}) \mid \\
& \{p_1, \ldots, p_\ell\} = \{p \mid (p, t) \in F\},\ \{p'_1, \ldots, p'_{\ell'}\} = \{p \mid (t, p) \in F\}, \\
& m = \max(\ell, \ell'),\ I_i = c(p_i, t) \text{ if } 1 \le i \le \ell \text{ else } I_i = [0, \infty), \\
& p_i = p_{capacity} \text{ if } \ell < i \le m,\ p'_i = p_{capacity} \text{ if } \ell' < i \le m\}
\end{aligned}$$

and we define $\max(t) \overset{\text{def}}{=} \max(|^\bullet t|, |t^\bullet|)$. Note that the max operator with two arguments is the classical maximum of two numbers.

The intuition is that $\mathit{Pairing}(t)$ fixes the paths from input to output places on which the tokens travel when firing the transition t, and it also remembers the associated time intervals and the type of the path (*tarc* for transport arcs and *normal* for the standard arcs that reset the ages of produced tokens). Observe that for the example net in Fig. 3 where $\max(t) = 3$ a possible pairing operator (used in the reduction) looks like $\mathit{Pairing}(t) = \{(p_0, [3, 7), p_1, \mathit{normal}), (p_1, [0, 9], p_2, \mathit{tarc}), (p_{capacity}, [0, \infty), p_3, \mathit{normal})\}$. Moreover, by $p \xrightarrow{I} t \longrightarrow p'$ we shall abbreviate the presence of an arc from p to t with the time interval I and an arc from t to p'; the type of the arcs (normal or transport) will be clear from the context. The translation is given in Alg. 1.

Notice that Alg. 1 for an input net $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$ creates an output net $N' = (P', T', F', c', F_{tarc}{}', c_{tarc}{}', \iota')$ such that

- $|P'| \le |P| + 2 + 4(|F| + 2|F_{tarc}|)$,

```
Algorithm 1: Translation from k-bounded TAPN to TAPN of degree 2
Input: A k-bounded TAPN $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$ with marking $M_0$.
Output: A TAPN $N' = (P', T', F', c', {F_{tarc}}', {c_{tarc}}', \iota')$ of degree 2 and $M'_0$.
begin
    $P' := P \cup \{p_{lock}, p_{capacity}\} \cup \{p_h(t^i) \mid t \in T,\ 1 \leq i < \max(t)\}$
               $\cup \{p(t^i_{in}), p(t^i_{out}) \mid t \in T,\ 1 \leq i < \max(t)\}$
    $T' := \{t^i_{in}, t^i_{out} \mid t \in T,\ 1 \leq i < \max(t)\} \cup \{t^{\max(t)}\}$
    $\iota'(p) := \begin{cases} \iota(p) & \text{if } p \in P \\ [0,0] & \text{if } p \in \{p(t^i_{in}), p(t^i_{out}) \mid t \in T,\ 1 \leq i < \max(t)\} \\ [0,\infty) & \text{otherwise} \end{cases}$
    forall $t \in T$ do
        $i := 1$
        while $|Pairing(t)| > 1$ do
            Remove some $(p, I, p', type)$ from $Pairing(t)$ and add arcs
            $p \xrightarrow{I} t^i_{in} \longrightarrow p_h(t^i)$ and $p_h(t^i) \xrightarrow{[0,\infty)} t^i_{out} \longrightarrow p'$ of type $type$.
            $i := i + 1$
        Let $\{(p, I, p', type)\} := Pairing(t)$; add arcs $p \xrightarrow{I} t^i \longrightarrow p'$ of type $type$.
        Add normal arcs $p_{lock} \xrightarrow{[0,\infty)} t^1_{in} \longrightarrow p(t^1_{in})$ and $p(t^1_{out}) \xrightarrow{[0,\infty)} t^1_{out} \longrightarrow p_{lock}$.
        Add normal arcs $p(t^i_{in}) \xrightarrow{[0,\infty)} t^{i+1}_{in} \longrightarrow p(t^{i+1}_{in})$ for $1 \leq i < \max(t) - 1$.
        Add normal arcs $p(t^{\max(t)-1}_{in}) \xrightarrow{[0,\infty)} t^{\max(t)} \longrightarrow p(t^{\max(t)-1}_{out})$.
        Add normal arcs $p(t^{i+1}_{out}) \xrightarrow{[0,\infty)} t^{i+1}_{out} \longrightarrow p(t^i_{out})$ for $1 \leq i < \max(t) - 1$.
    $M'_0(p) = \begin{cases} M_0(p) & \text{if } p \in P \\ \{0\} & \text{if } p = p_{lock} \\ \{\underbrace{0, \ldots, 0}_{k-|M_0|}\} & \text{if } p = p_{capacity} \\ \emptyset & \text{otherwise} \end{cases}$
end
```

- $|T'| \leq 2(|F| + 2|F_{tarc}|)$, and
- $|F'| + |{F_{tarc}}'| \leq 8(|F| + 2|F_{tarc}|)$.

Hence the translation causes only a linear growth in the size.

We shall now introduce a precise relationship between markings in a given marked k-bounded TAPN (N, M_0) and markings in the TAPN (N', M'_0) constructed by Alg. 1.

A marking $M' \in \mathcal{M}(N', M'_0)$ is called *stable* iff $|M'(p_{lock})| = 1$. Let $M \in \mathcal{M}(N, M_0)$ and $M' \in \mathcal{M}(N', M'_0)$. We say that M and M' correspond to each other, written $M \equiv M'$, if and only if

$$M'(p) = \begin{cases} M(p) & \text{if } p \in P \\ \{x\} & \text{if } p = p_{lock} \\ \{x_1, \ldots, x_{k-|M|}\} & \text{if } p = p_{capacity} \\ \emptyset & \text{otherwise} \end{cases} \quad \text{for some } x, x_1, \ldots, x_{k-|M|} \in \mathbb{R}^{\geq 0} \,.$$

Remark 3.1. *Note that for a given marking M there may be many markings M' such that $M \equiv M'$, but whenever $M \equiv M'$ then M' is stable. Intuitively, the age of the token x in the place p_{lock} represents the time that has elapsed since the last transition firing.*

Lemma 3.2. *Let (N, M_0) be a marked k-bounded TAPN and let (N', M'_0) be the marked TAPN of degree 2 constructed by Alg. 1. Let $M \in \mathcal{M}(N, M_0)$ and $M' \in \mathcal{M}(N', M'_0)$ such that $M \equiv M'$.*

1. *If $M \xrightarrow{t} M_1$ then $M' \longrightarrow^* M'_1$ such that $M_1 \equiv M'_1$ and the sequence by which M'_1 is reached from M' contains only discrete transitions.*

2. *If $M \xrightarrow{d} M_1$ then $M' \xrightarrow{d} M'_1$ such that $M_1 \equiv M'_1$.*

3. *If $M' \longrightarrow^* M'_1$, M'_1 is stable, none of the intermediate markings between M' and M'_1 are stable, and the first transition is not a time delay, then $M \xrightarrow{t} M_1$ for some $t \in T$ such that $M_1 \equiv M'_1$.*

4. *If $M' \xrightarrow{d} M'_1$ then $M \xrightarrow{d} M_1$ such that $M_1 \equiv M'_1$.*

Proof. We shall argue for the four cases while implicitly assuming $M \equiv M'$.

1. Let $M \xrightarrow{t} M_1$. We have to show that $M' \longrightarrow^* M'_1$ via a series of discrete transition firings such that $M_1 \equiv M'_1$. Clearly the tokens consumed in N when t is fired in the marking M are also present in M' in identically named places of the net N'. By definition of $\equiv$ their ages are moreover the same. From the construction of the net N' we can see that firing the series of transitions $t^1_{in}, t^2_{in}, \ldots, t^{max(t)-1}_{in}, t^{max(t)}$, $t^{max(t)-1}_{out}, t^{max(t)-2}_{out}, \ldots, t^1_{out}$ will bring the net N' into a stable marking M'_1 such that $M_1 \equiv M'_1$. Notice that during this firing sequence we may consume tokens from the capacity place (if $|{}^\bullet t| < |t^\bullet|$), or deposit tokens to the capacity place (if $|{}^\bullet t| > |t^\bullet|$),

however, the total number of tokens in N' remains constant and ages of tokens in the capacity place do not influence the firing of the above mentioned sequence because all arcs from $p_{capacity}$ have the interval $[0, \infty)$ associated to them.

2. Let $M \xrightarrow{d} M_1$. We know that M' is a stable marking hence there are no tokens in any of the newly added places except for p_{lock}. Because p_{lock} has the invariant $[0, \infty)$, all time delays allowed in M are allowed in M' and by $M' \xrightarrow{d} M'_1$ we clearly get $M_1 \equiv M'_1$.

3. Let $M' \longrightarrow^* M'_1$ such that M'_1 is the first stable marking in the firing sequence. Clearly, in any unstable marking during this sequence, there is exactly one token in some of the newly added places of the form $p(t^i_{in})$ or $p(t^i_{out})$. These places have an associated invariant $[0, 0]$ which means no time delays are possible. Hence any such sequence consists of exclusively discrete transition firings. We want to show that $M \xrightarrow{t} M_1$ for some t such that $M_1 \equiv M'_1$. By the construction of N' we observe that any firing sequence ending in a stable marking has the form: $t^1_{in}, t^2_{in}, \ldots, t^{max(t)-1}_{in}, t^{max(t)}, t^{max(t)-1}_{out}, t^{max(t)-2}_{out}, \ldots, t^1_{out}$. Notice that such a sequence cannot interleave with a transition sequence corresponding to any other transition because there is no token in the place p_{lock} in all the intermediate markings. From the construction of the net we can now easily see that $M \xrightarrow{t} M_1$ such that $M_1 \equiv M'_1$ by selecting the same ages of tokens in places from ${}^\bullet t$ as those consumed in the firing of the transitions $t^1_{in}, t^2_{in}, \ldots, t^{max(t)-1}_{in}, t^{max(t)}$.

4. Let $M' \xrightarrow{d} M'_1$. By the construction the same time delay is possible also from M such that $M \xrightarrow{d} M_1$ (all invariants in N are present also in N'). Clearly, $M_1 \equiv M'_1$.

□

We now describe how to translate queries. Formulae of the form $\psi = \mathsf{EF}\, \varphi$ are translated into $\psi' = \mathsf{EF}(\varphi \wedge p_{lock} = 1)$, and formulae of the form $\psi = \mathsf{AG}\, \varphi$ are translated into $\psi' = \mathsf{AG}(\varphi \vee p_{lock} = 0)$.

Theorem 3.3. *Let (N, M_0) be a marked k-bounded TAPN and let ψ be a formula of the form $\mathsf{EF}\, \varphi$ or $\mathsf{AG}\, \varphi$. Let (N', M'_0) be the marked TAPN of degree 2 constructed by Alg. 1 and let ψ' be the formula defined above. Then $M_0 \models_N \psi \iff M'_0 \models_{N'} \psi'$.*

Proof. Notice that the translation returns M'_0 such that $M_0 \equiv M'_0$. We will use this fact implicitly in the arguments to follow. First, we prove the theorem for the EF operator.

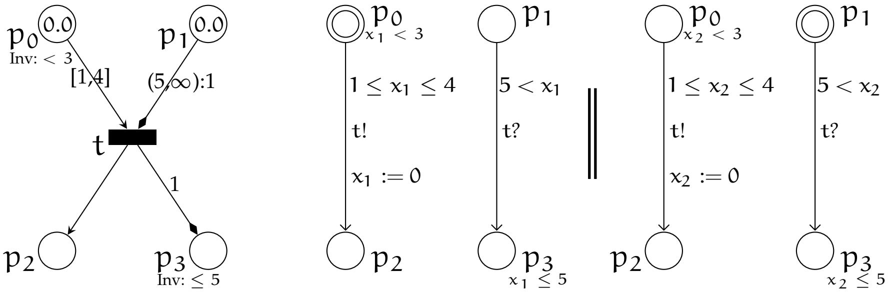


Figure 4: An example of the translation from TAPN to NTA.

"$\Rightarrow$" (EF): Let $M_0 \models \mathsf{EF}\,\varphi$, which means that $M_0 \longrightarrow^* M$ such that $M \models \varphi$. By repeatedly using Lemma 3.2 we get that $M'_0 \longrightarrow^* M'$ such that $M \equiv M'$, which gives that $M' \models \varphi$. Because M' is stable we get $M' \models \varphi \wedge p_{lock} = 1$ and this implies that $M'_0 \models \mathsf{EF}(\varphi \wedge p_{lock} = 1)$.
"$\Leftarrow$" (EF): Let $M'_0 \models \mathsf{EF}(\varphi \wedge p_{lock} = 1)$. This means that $M'_0 \longrightarrow^* M'$ such that M' is stable and $M' \models \varphi$. By repeatedly using Lemma 3.2 we get that $M_0 \longrightarrow^* M$ such that $M \equiv M'$, which means that $M \models \varphi$ and hence $M_0 \models \mathsf{EF}\,\varphi$.

The validity of the theorem for the AG operator follows for the definition and the above proved facts about EF as follows: $M_0 \models \mathsf{AG}\,\varphi \iff M_0 \not\models \mathsf{EF}\,\neg\varphi \iff M'_0 \not\models \mathsf{EF}(\neg\varphi \wedge p_{lock} = 1) \iff M'_0 \not\models \mathsf{EF}\,\neg(\varphi \vee p_{lock} \neq 1) \iff M'_0 \not\models \mathsf{EF}\,\neg(\varphi \vee p_{lock} = 0) \iff M'_0 \models \mathsf{AG}(\varphi \vee p_{lock} = 0)$. □

3.2 From TAPN of Degree 2 to Networks of Timed Automata

We can now assume a given net of degree 2 produced by our previous translation and we will continue with a construction of a network of timed automata. The idea of the translation is to represent each token in the net by a single timed automaton with one local clock, and to simulate a transition firing by a handshake synchronisation on a channel named after the transition.

The intuition is described on an example in Fig. 4. We can see that every place in the net gives rise to an identically named location in the parallel component corresponding to a given token, while all invariants are carried over. Time intervals on arcs are naturally transformed into guards and the local clocks of each parallel component are reset if and only if the transitions correspond to normal arcs. In fact, the timed automata for all tokens in the net are identical, except for their initial locations that are determined by the placement of tokens in the initial marking and the names of local clocks. The full translation is given in Alg. 2. For a TAPN of degree 2 with k tokens we hence create k parallel components, each of them of a proportional size to the input net.

Algorithm 2: Algorithm for translation of TAPN of degree 2 to NTA

```
Input: A TAPN $N = (P, T, F, c, F_{tarc}, c_{tarc}, \iota)$ of degree 2 and a marking $M_0$.
Output: An NTA $P_{TA} = A_1|A_2|\dots|A_{|M_0|}$ where $A_i = (L, \mathcal{A}ct, C, \longrightarrow_i, \iota_i, \ell_i^0)$.
begin
    $L := P$; $\mathcal{A}ct := \{t!,\ t? \mid t \in T\}$; $C := \{x_1, x_2, \dots, x_{|M_0|}\}$
    forall $t \in T$ do
        Let $\{(p_1, I_1, p'_1, type_1), (p_2, I_2, p'_2, type_2)\} := Pairing(t)$.
        for $i := 1$ to $|M_0|$ do
            Add $p_1 \xrightarrow{x_i \in I_1, t!, R}_i p'_1$ s.t. $R = \{x_i\}$ if $type_1 = normal$ else $R = \emptyset$.
            Add $p_2 \xrightarrow{x_i \in I_2, t?, R}_i p'_2$ s.t. $R = \{x_i\}$ if $type_2 = normal$ else $R = \emptyset$.
    $i := 1$; forall $p \in P$, forall $Token \in M_0(p)$ do $\ell_i^0 := p$; $i := i + 1$;
    for $i := 1$ to $|M_0|$ do forall $p \in P$ do $\iota_i(p)(x_i) := \iota(p)$
end
```

As for the first translation, we shall define a correspondence relation $\equiv$ between markings in the net and configurations of the constructed network of timed automata. Let $M = \{(p_1, r_1), (p_2, r_2), \cdots, (p_k, r_k)\}$ be a marking a TAPN of degree 2 and let $s = (l_1, \cdots, l_k, \nu)$ be a configuration of the constructed NTA. We write $M \equiv s$ if and only if for some permutation $\{j_1, j_2, \cdots, j_k\} = \{1, 2, \cdots, k\}$ we have $p_i = l_{j_i}$ and $\nu(x_{j_i}) = r_i$ for all i, $1 \le i \le k$.

Lemma 3.4. *Let (N, M_0) be a marked TAPN of degree 2. Let P_{TA} be the NTA constructed from (N, M_0). Let $M \in \mathcal{M}(N, M_0)$ and let s be a reachable configuration of P_{TA} such that $M \equiv s$.*

1. *If $M \xrightarrow{t} M'$ then $s \xrightarrow{t} s'$ and $M' \equiv s'$.*
2. *If $M \xrightarrow{d} M'$ then $s \xrightarrow{d} s'$ and $M' \equiv s'$.*
3. *If $s \xrightarrow{t} s'$ then $M \xrightarrow{t} M'$ and $M' \equiv s'$.*
4. *If $s \xrightarrow{d} s'$ then $M \xrightarrow{d} M'$ and $M' \equiv s'$.*

Proof. Let $s \in Conf(P_{TA})$ and $M \in \mathcal{M}(N, M_0)$ such that $M \equiv s$. This means that for each token in place p of age r (in the marking M) there is a parallel component A_i in the network of timed automata such that its current location is p and the value of the clock x_i is r, and vice versa.

1. Let $M \xrightarrow{t} M'$. Because t has exactly two input places, say p_0 and p_1, which contain tokens with ages that satisfy the invariants on arcs and $M \equiv s$, we can find two parallel components A_j and A_k in P_{TA} such that their current locations in s are p_0 and p_1, and their local clocks x_j and x_k represent the ages of the two tokens in p_0 and p_1 consumed when firing t. Because these components share the same time constraints as are the time intervals on the arcs in the Petri net, they can synchronize on the channel t and produce a state s' where clearly $M' \equiv s'$.

2.,4. Due to the construction of P_{TA} (all invariants are simply overtaken) any time delays allowed in M are also allowed in s and vice versa.

3. Let $s \xrightarrow{t} s'$ via a synchronization of two parallel components A_j and A_k on a channel t. Due to the construction of P_{TA} we can see that this can happen only if the transition t can be fired in M. Now let $M \xrightarrow{t} M'$ such that the firing of t involves tokens in ${}^\bullet t$ that correspond to the parallel components A_j and A_k. Clearly, $M' \equiv s'$.

□

Let ψ be a formula of our logic. By ψ' we denote a formula where atomic Petri net propositions of the form $(p \bowtie n)$ are replaced with propositions $(\#p \bowtie n)$ in the network of timed automata.

Theorem 3.5. *Let (N, M_0) be a marked TAPN of degree 2 and let ψ be a formula of the form $\mathsf{EF}\,\varphi$, $\mathsf{AG}\,\varphi$, $\mathsf{EG}\,\varphi$ or $\mathsf{AF}\,\varphi$. Let P_{TA} be an NTA constructed by Alg. 2 with the initial configuration $s_0 = (\ell_1^0, \ell_2^0, \ldots, \ell_{|M_0|}^0, \nu_0)$ and let ψ' be the formula defined above. Then $M_0 \models_N \psi \iff s_0 \models_{P_{TA}} \psi'$.*

Proof. Notice that the correspondence relation $\equiv$ is in fact a timed bisimulation and moreover $M \equiv s$ means that $M \models_N \varphi$ iff $s \models_{P_{TA}} \varphi'$ for every φ which is a Boolean combination of atomic propositions in the Petri net and φ' is the translated formula where every occurrence of $(p \bowtie n)$ is replaced with $(\#p \bowtie n)$. Because timed bisimilarity preserves TCTL model checking (and hence also our logic) and the atomic propositions do not distinguish between configurations related by the correspondence relation $\equiv$, we have established the validity of the theorem. □

3.3 Final Remarks

In a summary, for safety properties (EF and AG) we provided a translation from bounded timed-arc Petri nets to networks of timed automata by combining Theorem 3.3 and Theorem 3.5. For liveness properties we achieved such a translation for nets of degree 2 by using Theorem 3.5. Even though many net models of real-systems are already of degree 2 or can be easily modified so that Theorem 3.5 becomes applicable, for the nets where it is necessary to have transitions with more than two input places other translations have to be designed. The main obstacle is that the translation presented in Alg. 1 introduces new time-locks which cannot be distinguished from the time-locks in the original net.

4 Experiments

We shall now report on two experiments testing the efficiency of the translations from Section 3. The translations were implemented in the tool TAPAAL [10] and the models used in the experiments can be downloaded at `www.tapaal.net`. The reported running times were measured on a Dell PowerEdge 2950, with a 2.5 GHz, Dual Core Intel Xeon 5420 processor and 32GB ram. Notice, however, that UPPAAL utilises only one core and addresses at most 4GB of RAM.

4.1 Fischer's Protocol for Mutual Exclusion

Fischer's protocol [13] for ensuring mutual exclusion for a number of timed processes is a well-known protocol used for testing performances of tools. It is an easily scalable algorithm and provides a suitable case study for our translation because it requires that every process has its own independent clock. In other tools for Petri nets, such as TINA [6] and ROMEO [17], it is inconvenient to model Fischer's protocol as here clocks are usually associated to transitions and hence the number of processes is a priory fixed. One has to necessarily modify the static structure of the net when more processes need to be considered. In our approach we only need to add extra tokens to the same underlying net in order to increase the number of processes. The timed-arc Petri net model of Fischer's protocol is taken from [2] and it is available as an example in the TAPAAL distribution.

# Processes	Time in seconds			Speed-up
	UPPAAL	TAPAAL		
	Default	Standard	Optimised	
50	7.8	9.8	4.5	73 %
60	18.7	21.1	8.9	110 %
70	40.5	42.0	17.0	138 %
80	78.2	75.7	30.4	157 %
90	138.7	136.1	49.7	179 %
100	235.9	206.4	77.3	205 %
150	31m	22m	8m	293 %
200	2h 22m	1h 25m	29m	393 %
300	22h 7m	10h 6m	3h 24m	549 %
400	–	–	14h 9m	–

Figure 5: Fischer's Protocol with Symmetry Reduction Turned On

We verified the correctness of the Fischer's protocol for a different number of processes. The results were compared with the verification times of the UPPAAL model of Fischer's protocol from the UPPAAL demo folder. The experiments were run with symmetry reduction turned on, both in TAPAAL and UPPAAL, and with the default search options. The verification results are presented in Fig. 5. Here TAPAAL standard reduction is the one from Section 3 and TAPAAL optimised reduction replaces the locking token in the net with a global Boolean variable in order to reduce the size of the produced UPPAAL templates. Details of the optimised translation are given in the appendix. The speed-up column compares the running times between the UPPAAL model and the model produced by the TAPAAL optimised reduction. Even though the number of explored states in the network produced by TAPAAL is about two times as many as the ones in the native UPPAAL model, the verification times are significantly shorter. The reason for this seems to be the fact that the sizes of zones in the TAPAAL produced model are smaller and hence the expensive operation of zone inclusion checking is faster in our approach.

# Messages	Time in seconds		
	UPPAAL	TAPAAL	
	Default	Standard	Optimised
12	7.7	2.4	8.9
13	17.6	3.6	21.8
14	19.4	5.1	62.6
15	136	7.3	192.2
16	10m	10.1	11m
17	32m	13.7	35m
20	18h 37m	32.5	19h 34m
30	–	5m	–
40	–	29m	–
50	–	1h 52m	–

Figure 6: Alternating Bit Protocol with Symmetry Reduction Turned On

4.2 Alternating Bit Protocol

Alternating Bit Protocol (ABP) [4] is a simple instance of a sliding window protocol with windows of size one. ABP is an unbounded protocol, since in communication between a sender and a receiver, an arbitrary number of messages (each with an individual timestamp) can be in transfer via a lossy communication media. Details of the model are given in the appendix.

In Fig. 6 we present the verification results for a fixed number of messages in the system. The translation described in Section 3 allows us to verify the protocol for up to 50 messages in less than two hours. For comparison we created a UPPAAL model of ABP where all messages in the system are symmetric (see the appendix). Notice that the standard translation, contrary to the results from Fischer's protocol, is considerately faster than the optimised translation, described in the appendix, which is comparable with the native UPPAAL model we created. The reason seems to be the same as in Fischer's protocol: even though the number of stored and explored states is about twice as large, the zones are less complex and hence the inclusion check is faster.

In the future work we plan to study in detail this phenomenon and optimise the reductions (perhaps depending on the analysis of the concrete net) in order to achieve a further improvement in verification of TAPN models.

5 Conclusion

We studied timed-arc Petri nets extended with invariants on places and with transport arcs—new features that allow for a more convenient modelling of systems. The extended Petri net model with a bounded number of tokens was translated to networks of timed automata, preserving logical queries formulated in a subset of CTL. We employed a novel translation where a new component in the timed automata network was created for each token in the net.

The presented approach for verification of bounded timed-arc Petri nets is efficient as documented on two case studies modelled in the tool TAPAAL and in fact we outperform in the verification times of native UPPAAL models. We kept the considered logic simple and it is essentially identical with the presently implemented logical queries in UPPAAL. Nevertheless, we sketched that verification of TCTL queries and reasoning about the exact ages of tokens can be done by simple encoding tricks.

One cannot hope for a fully automatic verification of unbounded timed-arc Petri nets as, for example, the reachability problem becomes already undecidable [16]. On the other hand, the chosen reduction strategy enables one to further extend the bounded model with e.g. urgent transitions, priorities, cost, probability and game semantics, requiring only minor changes in the proposed reductions. In the future work we shall address these issues.

Acknowledgments. We would like to thank Alexandre David, Krishna Prasad Gundam and Ye Tian for their comments and suggestions. We also thank the anonymous reviewers for their feedback.

References

[1] P.A. Abdulla, P. Mahata, and R. Mayr. Dense-timed Petri nets: Checking zenoness, token liveness and boundedness. *Logical Methods in Computer Science*, 3(1):1–61, 2007.

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

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

[4] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Commun. ACM*, 12(5):260–261, 1969.

[5] B. Berthomieu, F. Peres, and F. Vernadat. Bridging the gap between timed automata and bounded time Petri nets. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *LNCS*, pages 82–97. Springer-Verlag, 2006.

[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*, 42(14):2741–2756, 2004.

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

[8] P. Bouyer, S. Haddad, and P.-A. Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. *Information and Computation*, 206(1):73–107, 2008.

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

[10] J. Byg, K.Y. Joergensen, and J. Srba. TAPAAL: Editor, simulator and verifier of timed-arc Petri nets. In *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09)*, volume 5799 of *LNCS*. Springer-Verlag, 2009. To appear.

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

[12] Kurt Jensen, Lars Kristensen, and Lisa Wells. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(3):213–254, 2007.

[13] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, 1987.

[14] W. Penczek and A. Pólrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*. Springer-Verlag, 2006.

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

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

[17] Ch. Seidner, G. Gardey, D. Lime, M. Magnin, and O. Roux. Romeo: A tool for time Petri net analysis. `http://romeo.rts-software.org/`.

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

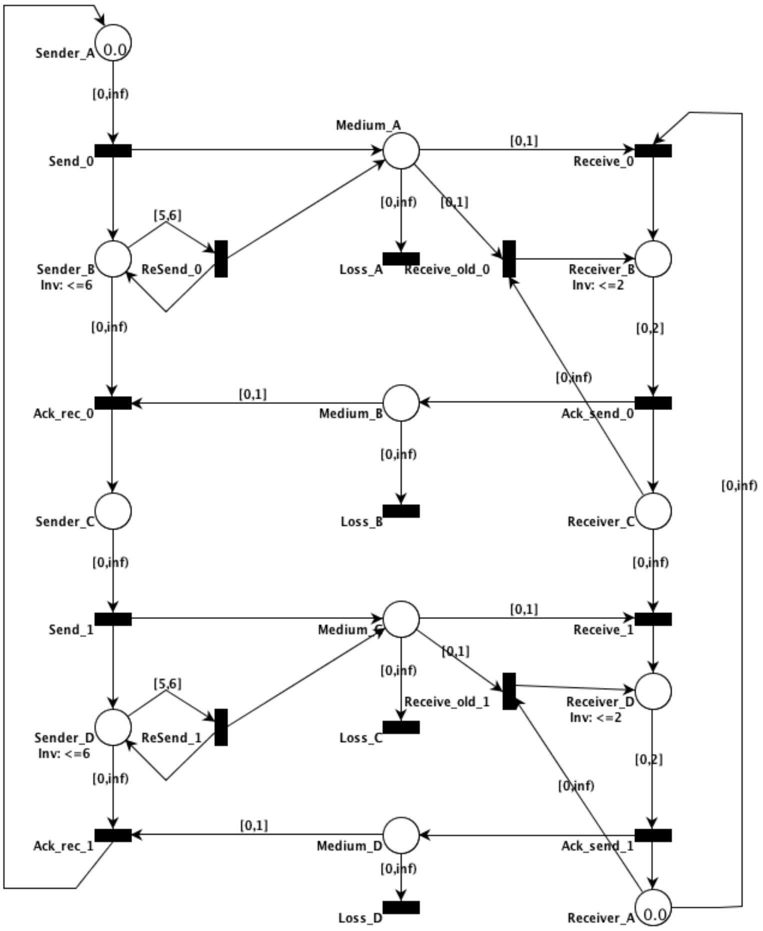
[19] J. Srba. Timed-arc Petri nets vs. networks of timed automata. In *Proceedings of the 26th International Conference on Application and Theory of Petri Nets (ICATPN 2005)*, volume 3536 of *LNCS*, pages 385–402. Springer-Verlag, 2005.

[20] J. Srba. Comparing the expressiveness of timed automata and timed extensions of Petri nets. In *6th Int. Conf. on Formal Modelling and Analysis of Timed Systems (FORMATS'08)*, volume 5215 of *LNCS*, pages 15–32. Springer, 2008.

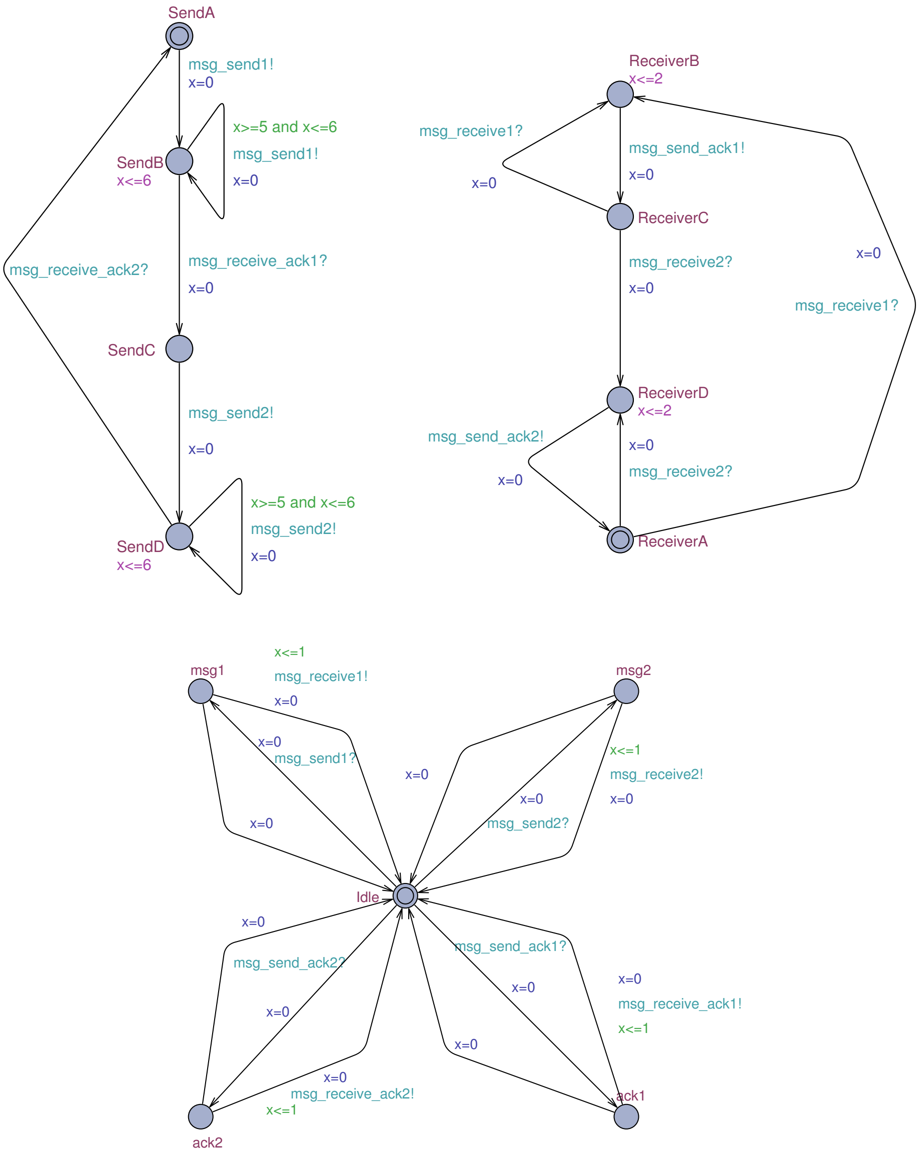
[21] UPPAAL. `www.uppaal.com`.

Appendix: Alternating Bit Protocol Modelling

The model of the alternating bit protocol as a TAPN with invariants is provided below. The number of extra tokens in the place $p_{capacity}$ (created during the translation) determines the largest number of messages that can at the same time appear in the protocol. The query that checks for violation of the synchronization between Sender and Receiver is: `EF (Sender_A=1 and Receiver_B=1) or (Sender_A=1 and Receiver_C=1) or (Sender_C=1 and Receiver_A=1) or (Sender_C=1 and Receiver_D=1).` As the protocol is correct, the query returns *false*.



The UPPAAL model of the alternating bit protocol (that we manually created in order to compare the performance with respect to the automated translation provided in TAPAAL), consists of three templates of Sender (upper left template), Receiver (upper right template) and Message (the template at the bottom). Each template has its private local clock x. The number of occurrences of the last template Message (all of them defined as symmetric ones) in the network determines the largest number of messages that can at the same time appear in the protocol. The verified query is: `E<> (Sender.SendA and Receiver.ReceiverB) or (Sender.SendA and Receiver.ReceiverC) or (Sender.SendC and Receiver.ReceiverA) or (Sender.SendC and Receiver.ReceiverD)`. The query returns *false* as the protocol is correct.



Appendix: Implementation of the Optimised Reduction

In this section we describe the ideas behind the implementation of the optimised reduction in the tool TAPAAL.

Recall the approach introduced in Section 3. Here the locking token takes care of two tasks: preventing the interleaving of different transition firings and determining the order of the firing of intermediate transitions.

In UPPAAL we can use a Boolean variable (let us call it `lock`), to prevent the interleaving. By applying this, we can in the cases where the input and output degree of a transition is less or equal to two eliminate the locking token. In this situation, one transition firing is simulated by a single synchronization between two templates. The optimised translation is shown in Figure 7 where the clock x is local to both templates. Clearly, the unreachable parts in the UPPAAL templates can be removed, we present them only for the clarity reasons.

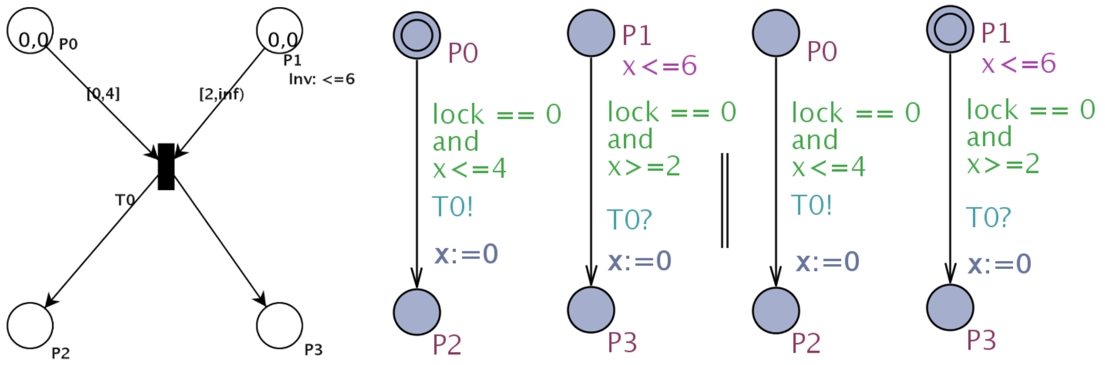


Figure 7: A Net of Degree 2 and the Produced UPPAAL Templates

When the net contains also transitions of degree-3 or higher, we still use the locking token in order to synchronize stepwise among several parallel components. The translation for this situation is shown on the TAPN in Figure 8 resulting in the NTA in Figure 9. As before, the unreachable parts in the UPPAAL templates can be easily removed and the clock x is local in all templates where it occurs.

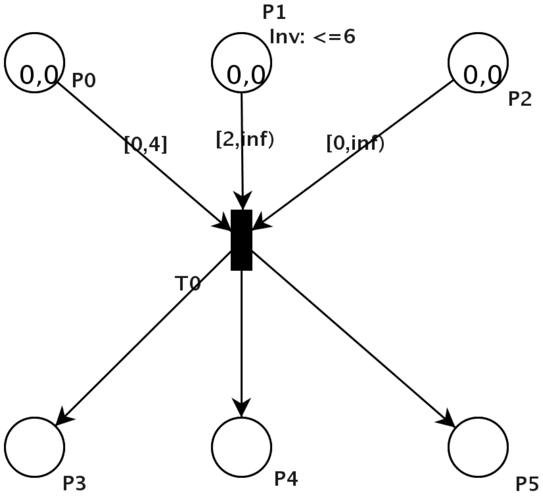


Figure 8: A Net of Degree 3

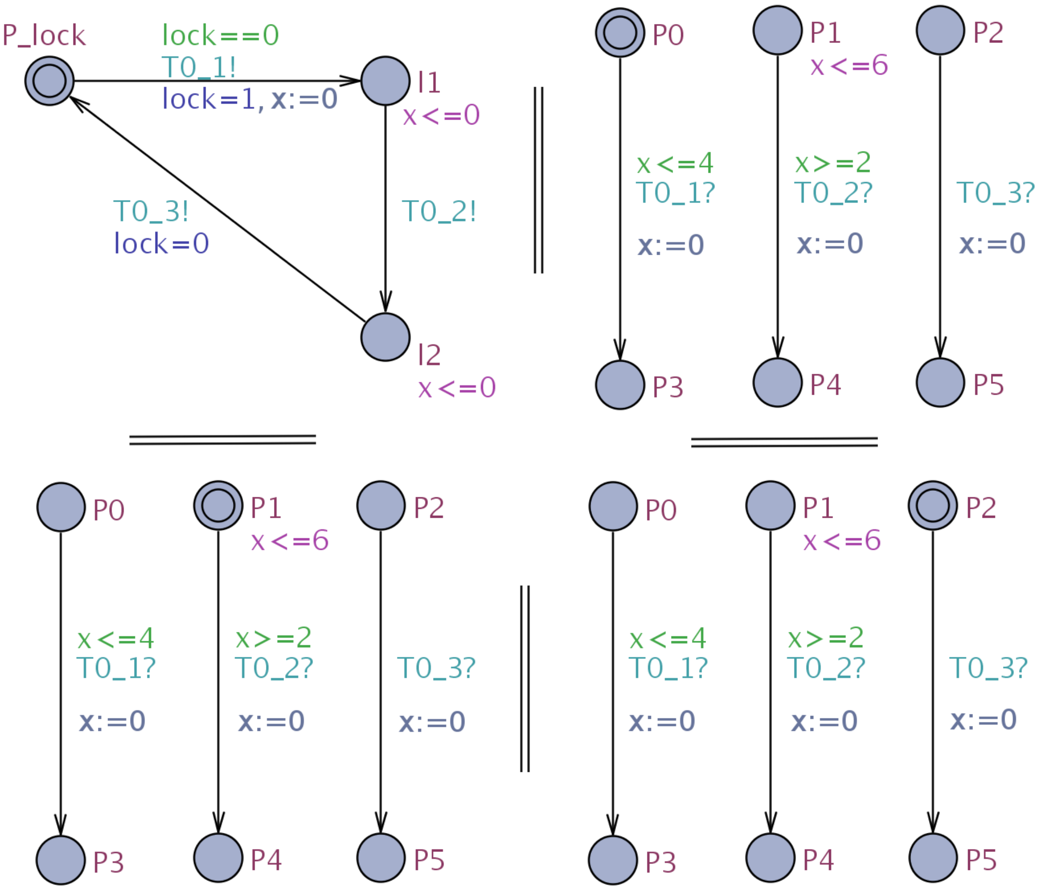


Figure 9: Four UPPAAL Templates Produced for the Net in Figure 8