Real-Time System Verification using P/T Nets

Roberto Gorrieri Glauco Siliprandi

Università di Bologna, Dipartimento di Matematica
Piazza di Porta San Donato 5, I - 40127 Bologna (Italy)
e-mail:{gorrieri,silipran}@cs.unibo.it

Abstract

Timed Nets are proposed to model the behavior of real-time systems. In our approach, net transitions are annotated by timing constraints, using finitely many real-valued *clocks*. A timed net accepts *timed words*, i.e. infinite sequences in which a real-valued time of occurrence is associated with each symbol. We study expressiveness, closure properties and decision problems of such nets. We also consider nondeterministic structures and acceptance conditions based on actions. The main construction of the paper is an algorithm for deciding the inclusion problem: this leads to automatic verification of real-time systems.

1 Introduction

There is a great variety of automata-based approaches to the specification and verification of real-time systems, depending mainly on the assumptions about the nature of time.

The use of *dense-time* domains (events may happen arbitrarily close to each other) as opposed to *discrete-time* has been advocated by, e.g., [8, 2]: it is not possible to impose a minimal time interval between two events in *hybrid* systems, which are composed of digital and analogical (hence, continuous) devices. Hence, dense-time is an important feature whenever one is interested in modeling such heterogeneous systems.

Among the models based on dense-time, particular interest has been stirred up by *Timed Automata*, proposed by Alur and Dill [3, 4, 5]. This model is essentially the timed version of finite-state ω-automata, hence recognizing *timed words* – infinite sequences in which a real-valued time is associated with each symbol. A timed word is recognized by one of such automata if, for instance, the set of those states passed through infinitely often is equal to a given set of accepting states. The behavior of a real-time system is modeled by a timed language $\mathcal{L}_{imp}$; since also the requirements the system must satisfy can be expressed as a timed language $\mathcal{L}_{spec}$, the problem of verifying that a system satisfies a certain property essentially reduces to the inclusion problem of the implementation timed language $\mathcal{L}_{imp}$ into the specification timed language $\mathcal{L}_{spec}$. Due to the decidability of the inclusion problem, Timed Automata have been profitably used for the automatic formal verification of real-time finite-state systems [4, 5].

Despite of their elegant characterization and strong properties, Timed Automata are not expressive enough to model systems with an infinite number of states. Here, we extend the approach of Alur and Dill to a more general class of automata, namely Place/Transition Petri Nets [11], where the number of tokens in each place can increase unboundedly. As the global state of a system is the collection of the tokens in the places, the number of global states of the system a P/T net represents is infinite.

Our approach follows an action based acceptance condition: a timed word is accepted iff the set of those transitions fired infinitely often belongs to a given family of sets of transitions. The choice of such an acceptance condition is due to the fact that the set of transitions of a (Timed) P/T Net is finite. This permits to prove some relevant decidability results: the problem of language emptiness and the problem of inclusion of a timed P/T net language in a (deterministic) timed regular language. These are at the base of automatic verification of real-time properties, expressed as Timed Automata, of real-time systems, expressed as Timed P/T Nets.

An example in Section 7 shows a simple real-time system which is modeled by a Timed P/T Net and which cannot be represented through a Timed Automaton, hence proving that the class of systems we can model is strictly larger. The final section introduces a simple example of application of our theoretical results.

Note: Due to space limit, most proofs are omitted or simply sketched; they can be found in the full version of this paper [9] and in [12].

2 ω-languages and ω-automata.

In this section we give some basic definitions concerning ω-words and their recognizing automata, namely finite state automata and Petri Nets. For more details see [13, 14].

Let Σ be a finite alphabet. Σ^* is the set of all finite sequences over Σ. An infinite sequence over Σ, also called ω-word, is a map $\sigma : I\!N^+ \to \Sigma$. Since σ_i denotes the ith symbol of σ, we also write $\sigma = \sigma_1\sigma_2\ldots$. The set of all ω-words over Σ is denoted by Σ^ω. If A is another alphabet, $h : \Sigma \to A$ and $\sigma \in \Sigma^\omega$, we use $h(\sigma)$ to denote the ω-word $h(\sigma_1)h(\sigma_2)\ldots \in A^\omega$. $L \subseteq \Sigma^\omega$ will be called a *ω-language*.

A property $P(n)$ holds infinitely often – denoted by $\exists^\infty n.P(n)$ – if $\forall m \in I\!N.\exists n > m.P(n)$ holds. We define $In(\sigma) \stackrel{\text{def}}{=} \{a \in \Sigma |\, \exists_i^\infty.\sigma_i = a\}$.

In the literature (see [13] for a survey on this topic) various types of finite state ω-automata have been studied: among these, we recall Büchi automata [7]. A Transition Table (TT for short) is a tuple $\mathcal{A} = (\Sigma, S, S_0, E)$, where Σ is an alphabet, S is a finite set of states, $S_0 \subseteq S$ is a set of start states, and $E \subseteq S \times S \times \Sigma$ is a set of transitions.

A *run* $r = (s_0, s_1, \sigma_1)(s_1, s_2, \sigma_2)\ldots \in E^\omega$ on an ω-word σ is such that $s_0 \in S_0$. The sequence of *reached states* in the run r is the ω-word $s_1s_2s_3\ldots \in S^\omega$, denoted by $St(r)$. TTs can be used to recognize ω-languages.

Definition 2.1 (Büchi Automata) A BA is a tuple $\mathcal{A} = (\Sigma, S, S_0, E, F)$ such that (Σ, S, S_0, E) is a TT and $F \subseteq S$. A run r is *accepted* iff $In(St(r)) \cap F \neq \emptyset$. The accepted ω-language is $\mathcal{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{\sigma \in \Sigma^\omega |$ there is a run on σ accepted by $\mathcal{A}\}$. The class of languages accepted by BAs, called *regular ω-languages*, is denoted by $\mathcal{BA}$. □

Now we define Petri nets and their infinite behavior by ω-languages (see [14] for more details).

Definition 2.2 (Place/Transition Petri Nets) A PTP is a tuple $N = (\Sigma, P, T, F, W, h, m_0)$ where Σ is an input alphabet, P is a finite set of *places*, T is a finite set of *transitions* (disjoint from P), $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*, $W : F \to I\!N^+$ is the *weight function*, $h : T \to \Sigma$ is the *labeling function*, $m_0 \in I\!N^P$ is the *initial marking* (the functions from P to $I\!N$, are called *markings* for N; such a set will be usually denoted by M and ranged over by m).

The *pre-set* of a transition t is ${}^\bullet t \stackrel{\text{def}}{=} \{p |\, (p, t) \in F\}$, and its *post-set* is $t^\bullet \stackrel{\text{def}}{=} \{p |\, (t, p) \in F\}$. N *has concession* in m iff $m(p) \geq W(p, t)$ for all $p \in {}^\bullet t$. If t has concession in m then t *fires* m to m', where $m'(p) = m(p) - W(p, t)$ if $p \in {}^\bullet t \setminus t^\bullet$, $m'(p) = m(p) + W(t, p)$ if $p \in t^\bullet \setminus {}^\bullet t$, $m'(p) = m(p) - W(p, t) + W(t, p)$ if $p \in {}^\bullet t \cap t^\bullet$ and $m'(p) = m(p)$ if $p \in P \setminus ({}^\bullet t \cup t^\bullet)$. This relation on $M \times T \times M$ is the *firing relation*, denoted by $m\ (t\rangle m'$.

We say that $r \in T^\omega$ is a run of N over $\sigma \in \Sigma^\omega$ iff $h(r) = \sigma$ and there are $m_1, m_2, \ldots \in M$ such that $\forall i \in I\!N.m_i\ (r_{i+1}\rangle m_{i+1}$. $m_0, m_1, \ldots$ is the *sequence of markings associated to* r. □

In [6] PTPs are provided with acceptance conditions based on the transitions that are fired infinitely often in a run. Among the various notions proposed there, we consider the weaker one, i.e. the one for which the class of accepted languages is the largest.

Definition 2.3 (Action Based PTP) An ABP is a tuple $N = (\Sigma, P, T, F, W, h, m_0, \mathcal{F})$ where $(\Sigma, P, T, F, W, h, m_0)$ is a PTP and $\mathcal{F} \subseteq \wp^+(T)$. A run r of N is accepted iff $\exists R \in \mathcal{F}.In(r) = R$. The class of ω-languages accepted by ABPs is denoted by $\mathcal{ABP}$. □

A relevant decidability result w.r.t. ABPs is the following

Theorem 2.4 *[15]The emptiness problem is decidable for $\mathcal{ABP}$.*

3 Timed Languages

In this section we present the timed languages, as introduced in [4]. An execution of a real-time system corresponds to a timed word over the alphabet of events; a timed word is a word σ, coupled with a time sequence τ.

Definition 3.1 (Time Sequence) A time sequence $\tau = \tau_1\tau_2\ldots$ is an infinite sequence of time values $\tau_i \in I\!R^{\geq 0}$ (the set of nonnegative reals) with $\tau_i > 0$, satisfying the following constraints:

Monotonicity: $\tau_i < \tau_{i+1}$ for all $i \geq 1$.

Progress: for every $t \in I\!R^{\geq 0}$ there is some $i \geq 1$ such that $\tau_i > t$.

For any time sequence $\tau = \tau_1\tau_2\ldots$ we assume $\tau_0 \stackrel{\text{def}}{=} 0$. □

The progress condition is introduced in order to avoid the Zeno paradox [1]: it will never be the case that infinitely many events happen in a bounded interval of time.

Definition 3.2 (Timed Words and Languages) A *timed word* over Σ is a pair (σ, τ) where $\sigma \in \Sigma^\omega$ and τ is a time sequence. Σ^t denotes the set of all timed words over Σ and is ranged over by ψ. A *timed language* is a set $L \subseteq \Sigma^t$. □

The *Untime* operation, for timed languages, discards the time values associated with the symbols, that is, it considers the projection of a timed word (σ, τ) on the first component.

Definition 3.3 (Untimed Language) Given a timed language L, we define $Untime[L] \stackrel{\text{def}}{=} \{\sigma \mid \exists\tau.(\sigma, \tau) \in L\}$, also denoted by $\pi_1(L)$ (the first projection on the elements of L). □

4 Timed Finite State Automata

In [3, 4, 5] the definition of finite state ω-automata is augmented, so that they accept timed languages. In this section we recall some results presented there and a new result of ours.
Transition tables are extended to *timed* transition tables so that they can read timed words. When executing a transition, the choice of the next state depends also upon the time of the input symbol w.r.t. the times of the previously read symbols. For this purpose a finite set of *clocks* is associated with each transition table. A clock can be set to zero simultaneously with the execution of a transition. At any instant, the value of a clock is equal to the time elapsed since the last time of reset. A *clock constraint* is associated with each transition. It is required that the transition may be taken only if the current values of the clocks satisfy this constraint.

Definition 4.1 (Clocks) For a set C of clock variables, the set $\Phi(C)$ of *clock constraints* is defined inductively by

$$\delta := c \leq q \mid q \leq c \mid \neg\delta \mid \delta \wedge \delta$$

where c is a *clock* in C and q is a nonnegative rational constant. A *clock interpretation* (CI for short) for C is a function $\nu : C \to I\!R^{\geq 0}$. We say that ν satisfies δ iff δ evaluates to true. ν_0

denotes the CI such that $\forall c \in C.\nu_0(c) \stackrel{\text{def}}{=} 0$. If $t \in I\!R^{\geq 0}$, then $\nu + t$ is the CI mapping every clock c to $\nu(c) + t$, and $t \cdot \nu$ the one assigning $t \cdot \nu(c)$ to each clock c. For $Y \subseteq C$, $\nu[t/Y]$ denotes the CI assigning t to each $c \in Y$ and agreeing with ν over the other clocks. $\square$

A Timed Transition Table (TTT for short) is a tuple $\mathcal{A} = (\Sigma, S, S_0, C, E)$, where C is a finite set of clocks, and $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ is the set of transitions. (s, s', a, y, δ) represents a transition from state s to state s' on input symbol a. The set $y \subseteq C$ gives the clocks to be reset when executing this transition, and δ is a clock constraint to be satisfied. A *run* of $\mathcal{A}$ on a timed word (σ, τ) is an infinite sequence of transitions $r = (s_0, s_1, \sigma_1, y_1, \delta_1)(s_1, s_2, \sigma_2, y_2, \delta_2)\ldots$ such that (i) $s_0 \in S_0$ and (ii) there are $\nu_1, \nu_2, \ldots$ CIs such that δ_{i+1} is satisfied by $\nu_i + (\tau_{i+1} - \tau_i)$ and $\nu_{i+1} \stackrel{\text{def}}{=} (\nu_i + (\tau_{i+1} - \tau_i))[0/y_{i+1}]$. We say that $\nu_0, \nu_1, \ldots$ is the *sequence of CIs associated* to r, and $St(r) \stackrel{\text{def}}{=} s_1 s_2 s_3 \ldots$ is the sequence of *reached states.*

Definition 4.2 (Timed Büchi Automata)
A TBA is a tuple $\mathcal{A} = (\Sigma, S, S_0, C, E, F)$, where (Σ, S, S_0, C, E) is a TTT, and $F \subseteq S$. A run r is accepted iff $In(St(r)) \cap F \neq \emptyset$. The accepted timed language is $\mathcal{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{\psi \in \Sigma^t \mid \mathcal{A}$ has an accepting run over $\psi\}$. The class of timed languages accepted by TBAs, called *timed regular languages*, is denoted by $\mathcal{TBA}$. $\square$

Definition 4.3 (Timed Muller Automata)
A TMA is a tuple $\mathcal{A} = (\Sigma, S, S_0, C, E, \mathcal{F})$, where (Σ, S, S_0, C, E) is a TTT, and $\mathcal{F} \subseteq \wp^+(S)$. A run r is accepted iff $In(St(r)) \in \mathcal{F}$. The set of timed languages accepted by TMAs is denoted by $\mathcal{TMA}$. $\square$

Theorem 4.4 *[4, 5]*
(i) $\mathcal{TBA} = \mathcal{TMA}$

(ii) *$\mathcal{TBA}$ is closed under union and intersection, but not under complementation.*

In order to define a class of timed languages closed under all boolean operations, deterministic timed transition tables are investigated. A TTT (Σ, S, S_0, C, E) is called *deterministic* (DTTT) if and only if $|S_0| = 1$ and for all $s \in S$, for all $a \in \Sigma$, for every pair of transitions of the form $(s, -, a, -, \delta_1)$ and $(s, -, a, -, \delta_2)$, we have that $\delta_1 \wedge \delta_2$ is unsatisfiable.

Definition 4.5 (Deterministic Timed Automata) A TBA (TMA) is *deterministic*–called DTBA (DTMA)–iff its TTT is deterministic. The class of timed languages accepted by DTBA (DTMA) is denoted by $\mathcal{DTBA}$ ($\mathcal{DTMA}$). $\square$

Theorem 4.6 *[4, 5]*
(i) $\mathcal{DTBA} \subset \mathcal{DTMA} \subset \mathcal{TBA}$

(ii) *$\mathcal{DTBA}$ is closed under union and intersection, but not under complementation.*

(iii) *$\mathcal{DTMA}$ is closed under union, intersection and complementation.*

We now study new acceptance conditions based on transitions rather than states. We define the *action based timed Büchi automata* and the *action based timed Muller automata*, and we prove that these have the same expressive power of TBAs and TMAs.

Definition 4.7 (Action Based TBA) An ABTBA is a tuple $\mathcal{A} = (\Sigma, S, S_0, C, E, F)$, where (Σ, S, S_0, C, E) is a TTT, and $F \subseteq E$ is the set of *accepting transitions*. A run r is accepted iff $In(r) \cap F \neq \emptyset$. The set of accepted timed languages is denoted by $\mathcal{ABTBA}$. $\square$

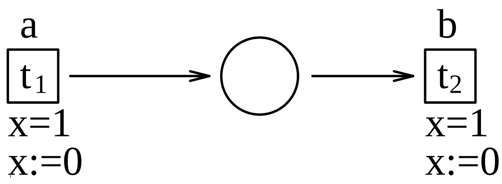

Figure 1: An example of timed P/T net.

Definition 4.8 (Action Based TMA) An ABTMA is a tuple $\mathcal{A} = (\Sigma, S, S_0, C, E, \mathcal{F})$, where (Σ, S, S_0, C, E) is a TTT, and $\mathcal{F} \subseteq \wp^+(E)$ specifies an *acceptance family*. A run r is accepted iff $In(r) \in \mathcal{F}$. The set of timed languages accepted by ABTMAs is denoted by $\mathcal{ABTMA}$. $\square$

In [12, 9] we prove the following relevant fact, which shows that no additional expressive power is obtained by considering these action based acceptance conditions.

Theorem 4.9 $\mathcal{ABTBA} = \mathcal{TBA} = \mathcal{TMA} = \mathcal{ABTMA}$.

In [9], however, we show that each one of the four acceptance conditions may be used to express specific real-time properties in more natural way.

5 Timed Place/Transition Nets

In this section we augment the definition of PTPs following the same idea illustrated in the previous section, so that they can recognize timed words.

Definition 5.1 (Timed P/T Net) A TP is a tuple $N = (\Sigma, P, T, F, W, h, m_0, C, c^c, c^r)$ where $N' = (\Sigma, P, T, F, W, h, m_0)$ is a PTP, C is a finite set of clocks, $c^c : T \to \Phi(C)$ gives the clock constraints associated to each transition, and $c^r : T \to 2^C$ gives the set of clocks to be reset when a transition is executed.
A *snapshot* for N is a couple (m, ν) such that m is a marking for N' and ν is a CI for C. The set of snapshots is denoted by Γ. A transition t of N *has concession* in (m, ν) for N iff t has concession in m for N' and ν satisfies $c^c(t)$. If t has concession in (m, ν), then t *fires* (m, ν) to (m', ν') where $m\ (t\rangle\ m'$ and $\nu' = \nu[0/c^r(t)]$. This relation on $\Gamma \times T \times \Gamma$ is the *timed firing relation* denoted by $(m, \nu)\ (t\rangle\ (m', \nu')$. A *run* of N over (σ, τ) is an infinite sequence $r = t_1 t_2 \ldots \in T^\omega$ such that $h(r) = \sigma$ and there are $(m_1, \nu_1), (m_2, \nu_2), \ldots$ snapshots for N such that $\forall i \in I\!N.(m_i, \nu_i + (\tau_{i+1} - \tau_i))\ (t_{i+1}\rangle(m_{i+1}, \nu_{i+1})$. $(m_0, \nu_0), (m_1, \nu_1), \ldots$ is the *sequence of snapshots associated* to r. $\square$

For example, consider the TP in Figure 1, where a and b are the labels for t_1 and t_2. In every run, in each moment of the execution, the number m of fired t_1 transitions is not less than the number n of executed t_2 transitions; indeed the place keeps exactly $m - n$ tokens. The timing constraints impose that a transition must be fired after each time unit, thus the first transition in the run fires at time 1, the second at time 2 and so on.
We provide TPs with an action based acceptance condition.

Definition 5.2 (Action Based TP) An ATP is a tuple $N = (\Sigma, P, T, F, W, h, m_0, C, c^c, c^r, \mathcal{F})$ where $(\Sigma, P, T, F, W, h, m_0, C, c^c, c^r)$ is a TP and $\mathcal{F} \subseteq \wp^+(T)$ is an *acceptance family*. A run r is accepted iff $\exists R \in \mathcal{F}.In(r) = R$. The accepted timed language is $\mathcal{L}(N) \stackrel{\text{def}}{=} \{\psi \in \Sigma^t |\ N$ has an accepting run over $\psi\}$. The class of accepted timed languages is denoted by $\mathcal{ATP}$. $\square$

Consider again the net in Figure 1, with acceptance family $\mathcal{F} \stackrel{\text{def}}{=} \{\{t_1\}, \{t_1, t_2\}\}$: it can be proved that every run is accepted and the recognized timed language is $\{(\sigma, \tau) \in \{a, b\}^t |\ \forall i \in I\!N.\#\{n \leq i |\ \sigma_n = a\} \geq \#\{n \leq i |\ \sigma_n = b\} \wedge \tau = 1\ 2\ 3 \ldots\}$.

6 Emptiness

In this section we develop an algorithm that, given an ATP accepting a timed language L, constructs an ABP recognizing $Untime[L]$ (in [4, 5] an analogous procedure was defined for TBAs). This important result allows us to state the decidability of the emptiness problem for timed languages in $\mathcal{ATP}$, and defines a limit to the expressive power of ATPs, in the sense that a timed language may be accepted by an ATP only if its $Untime$ is recognized by an ABP. We use this fact to prove the nonclosure of $\mathcal{ATP}$ under complementation.
First of all we want to prove that, in order to address our goal, it suffices to consider those nets in which just integer constants are used in clock constraints.

Definition 6.1 (Multiplied Clock Constraints) Given a TP N and $t \in \mathbb{R}^{\geq 0}$, N_t is the TP obtained by replacing each constant q, in each δ appearing in N, by $t \cdot q$. □

Lemma 6.2 *Consider a TP N, a timed word (σ, τ), and a positive rational t. Then r is a run of N over (σ, τ) iff r is a run of N_t over $(\sigma, t \cdot \tau)$.*

If we choose t to be the least common multiple of all constants appearing in the clock constraints of N, then the clock constraints in N_t use only nonnegative integers; furthermore, if N is an ATP then $Untime[\mathcal{L}(N)] = Untime[\mathcal{L}(N_t)]$; as a consequence, $\mathcal{L}(N) = \emptyset$ iff $\mathcal{L}(N_t) = \emptyset$. Thus, in the remainder of the section we assume that clock constraints use only nonnegative integers.
The problem of checking the emptiness of the timed language recognized by an ATP N can be more easily coped with if we show that this is equivalent to checking the emptiness of the language recognized by an ABP N'. This could be done, in principle, by adding places for clock interpretations: a snapshot $\langle m, \nu \rangle$ in N is simulated in N' by putting a token in the place corresponding to ν. Even if the CIs for N are infinite, we can define an equivalence relation $\cong$ on CIs, with a finite number of classes, and then add a new place for each of such classes. In [4, 5] $\cong$ has been defined in the case of TBAs.

Definition 6.3 (Clock Regions) Let q_c denote the largest integer q such that $c \leq q$ or $q \leq c$ is a subformula of some δ appearing in the net N; for any nonnegative real t, $fract(t)$ denotes the fractional part of t, and $\lfloor t \rfloor$ denotes the integer part of t; that is $t = \lfloor t \rfloor + fract(t)$.
$\nu \cong \nu'$ iff the following three conditions hold:

1. $\forall c \in C. \lfloor \nu(c) \rfloor = \lfloor \nu'(c) \rfloor \vee (\nu(c) > q_c \wedge \nu'(c) > q_c)$
2. $\forall c, d \in C. (\nu(c) \leq q_c \wedge \nu(d) \leq q_d) \Rightarrow$
$$(fract(\nu(c)) \leq fract(\nu(d)) \Leftrightarrow fract(\nu'(c)) \leq fract(\nu'(d)))$$
3. $\forall c \in C. \nu(c) \leq q_c \;\Rightarrow\; (fract(\nu(c)) = 0 \Leftrightarrow fract(\nu'(c)) = 0)$ □

It can be proved that $\cong$ is an equivalence relation. An equivalence class is called a *clock region* (CR for short). $[\nu]$ denotes the CR which ν belongs to. The set of CRs is ranged over by α.

Proposition 6.4 *Let $c \in C$, $\nu \cong \nu'$ and $t \in T$. Then*
1. ν satisfies $c^c(t)$ iff ν' satisfies $c^c(t)$

2. $\nu[0/c^r(t)] \cong \nu'[0/c^r(t)]$

3. $\nu(c) = 0$ iff $\nu'(c) = 0$ and $\nu(c) > q_c$ iff $\nu'(c) > q_c$.

Thus in the remainder of the section we shall say that $[\nu]$ satisfies $c^c(t)$ if ν satisfies $c^c(t)$, and we shall use $[v][0/c^r(t)]$ instead of $[\nu[0/c^r(t)]]$.
In [12, 9] an algorithm producing a representative CI to each clock region is provided: we use $\overline{\alpha}$ to denote the representative of class α. This construction permits to prove the following

Proposition 6.5

1. *The set of clock regions is finite.*

2. *If $\nu \cong \nu'$ and $t > 0$, then there exists $t' > 0$ such that $\nu + t \cong \nu' + t'$.*

Then a new place is introduced for each clock region. When a transition fires, it puts a token in a clock region place. This means that the CI reached just after the firing of the transition belongs to the clock region corresponding to that place. Moreover, the previous proposition states that, if $\nu \cong \nu'$, ν and ν' visit the same clock regions as time progresses.

Definition 6.6 (Time-successor) A CR α' is a *time-successor* of a CR α iff for each $\nu \in \alpha$ there exists $t > 0$ such that $\nu + t \in \alpha'$. □

In [12, 9] an algorithm which determines the representative of any time successor of a given CR is provided. Now we are ready to define the ABP recognizing $Untime[\mathcal{L}(N)]$.

Definition 6.7 (Region Net) Given a TP $N = (\Sigma, P, T, F, W, h, m_0, C, c^c, c^r)$ we define the PTP $\mathcal{R}(N) \stackrel{\text{def}}{=} (\Sigma, P', T', F', W', h', m_0')$ where

- $P' \stackrel{\text{def}}{=} P \cup \{\alpha |\, \alpha \text{ is a CR for } N\}$

- $T' \stackrel{\text{def}}{=} \{(t, \alpha, \alpha') |\, t \in T$ and there is an α'' time-successor of α, such that α'' satisfies $c^c(t)$ and $\alpha' = \alpha''[0/c^r(t)]\}$

- $F' \stackrel{\text{def}}{=} \{(p, (t, \alpha, \alpha')) |\, (p, t) \in F\} \cup \{((t, \alpha, \alpha'), p) |\, (t, p) \in F\} \cup \{(\alpha, (t, \alpha, \alpha')) |\, (t, \alpha, \alpha') \in T'\} \cup \{((t, \alpha, \alpha'), \alpha') |\, (t, \alpha, \alpha') \in T'\}$

- $W'(p, (t, \alpha, \alpha')) \stackrel{\text{def}}{=} \begin{cases} W(p, t) & \text{if } (p, t) \in F \\ 1 & \text{otherwise} \end{cases}$ $\quad W'((t, \alpha, \alpha'), p) \stackrel{\text{def}}{=} \begin{cases} W(t, p) & \text{if } (t, p) \in F \\ 1 & \text{otherwise} \end{cases}$

- $h'(t, \alpha, \alpha') \stackrel{\text{def}}{=} h(t)$

- $m_0'(p) \stackrel{\text{def}}{=} \begin{cases} m_0(p) & \text{if } p \in P \\ 1 & \text{if } p = [\nu_0] \\ 0 & \text{otherwise} \end{cases}$ □

In the net $\mathcal{R}(N)$, any reachable marking has one token in only one of the clock region places, while all the other CR places are empty.

Definition 6.8 ($\mathcal{R}(N)$ Markings) Let m be a marking of N. $\langle m, \alpha \rangle$ denotes the marking in $\mathcal{R}(N)$ such that $\langle m, \alpha \rangle(p) \stackrel{\text{def}}{=} m(p)$ if $p \in P$, $\langle m, \alpha \rangle(\alpha) \stackrel{\text{def}}{=} 1$, and $\langle m, \alpha \rangle(p) \stackrel{\text{def}}{=} 0$ otherwise. □

Now we want to establish a correspondence between each run in N and some run in $\mathcal{R}(N)$.

Definition 6.9 (Run Projection)
Let $r = t_1 t_2 \ldots$ be a run of N on (σ, τ) and $(m_0, \nu_0), (m_1, \nu_1), \ldots$ be the sequence of snapshots associated to r. We define the *run projection* of r to be $[r] \stackrel{\text{def}}{=} (t_1, [\nu_0], [\nu_1])(t_2, [\nu_1], [\nu_2]) \ldots$. □

Lemma 6.10 *$[r]$ is a run of the region net $\mathcal{R}(N)$.*

Proof: Let $i \in I\!N$. Then $[\nu_i + (\tau_{i+1} - \tau_i)]$ satisfies $c^c(t_{i+1})$ and $[\nu_{i+1}] = [\nu_i + (\tau_{i+1} - \tau_i)][0/c^r(t_{i+1})]$. Thus $[r]_{i+1} \in T'$. By induction on i, it can be proved that $\langle m_i, [\nu_i] \rangle([r]_{i+1}\rangle\, \langle m_{i+1}, [\nu_{i+1}] \rangle$. ■

Because of the progress condition, every clock in a run is either reset infinitely often, or from a certain time onwards it increases unboundedly.

Definition 6.11 (Progressive Run) A run of $\mathcal{R}(N)$, with associated sequence of markings $\langle m_0, [\nu_0]\rangle, \langle m_1, [\nu_1]\rangle, \ldots$, is *progressive* iff $\forall c \in C,\ \exists^\infty i.(\nu_i(c) = 0 \vee \nu_i(c) > q_c)$. □

Lemma 6.12 *If r is a run of N, then $[r]$ is a progressive run of $\mathcal{R}(N)$.*

Conversely, we now want to prove that, given a progressive run r' of $\mathcal{R}(N)$, a run r of N can be defined such that r' is its projection. We shall also define the sequence τ of time instants corresponding to firings in r. The following lemma ensures that τ progresses unboundedly.

Lemma 6.13 *Assume ν is a CI such that $\forall c \in C.(\nu(c) < \frac{1}{2} \vee \nu(c) > q_c)$, and assume α is a time-successor for $[\nu]$. Then there exists $t \geq \frac{1}{2}$ such that $\nu + t \in \alpha$.*

Proof: There must be $t' > 0$ such that $\nu + t' \in \alpha$, by definition of time-successor. In case $t' < \frac{1}{2}$, then it is easy to see that $\nu + \frac{1}{2} \cong \nu + t'$. ■

Lemma 6.14 *If $r' = (t_1, \alpha_0, \alpha_1)(t_2, \alpha_1, \alpha_2)\ldots$ is a progressive run of $\mathcal{R}(N)$ over σ, then there exist a time sequence τ and a run r of N over (σ, τ) such that $r' = [r]$.*

Proof: Let $\langle m_0, \alpha_0\rangle, \langle m_1, \alpha_1\rangle, \ldots$ be the sequence of markings associated to r'. We shall construct stepwise both r and τ.
Let $i \in I\!N$, and ν_i, τ_i be defined, such that $\nu_i \in \alpha_i$. Since ${r'}_{i+1} \in T'$, there must be an α time-successor of α_i such that α satisfies $c^c(t_{i+1})$, and $\alpha_{i+1} = \alpha[0/c^r(t_{i+1})]$. Thus there exists $t > 0$ such that $\nu_i + t \in \alpha$. Two cases are possible:

1. $\exists c \in C.\frac{1}{2} \leq \nu_i(c) \leq q_c$: then we define $\tau_{i+1} \stackrel{\text{def}}{=} \tau_i + t$; or
2. $\forall c \in C.\nu_i(c) < \frac{1}{2} \vee q_c < \nu_i(c)$: by Lemma 6.13 there exists $t' \geq \frac{1}{2}$ such that $\nu_i + t' \in \alpha$; hence $\tau_{i+1} \stackrel{\text{def}}{=} \tau_i + t'$

Moreover we define $\nu_{i+1} \stackrel{\text{def}}{=} (\nu_i + (\tau_{i+1} - \tau_i))[0/c^r(t_{i+1})]$. In both cases $\nu_i + (\tau_{i+1} - \tau_i) \in \alpha$, thus $(m_i, \nu_i + (\tau_{i+1} - \tau_i))\ (t_{i+1})(m_{i+1}, \nu_{i+1})$. Since $\nu_{i+1} \in \alpha_{i+1}$, the next τ_{i+2}, and ν_{i+2} can be defined as above, and so on. We prove that τ is unbounded in both cases:

1. Suppose $\exists^\infty_i.\exists c.\frac{1}{2} \leq \nu_i(c) \leq q_c$. Since C is finite, then $\exists c \in C.\exists^\infty_i.\frac{1}{2} \leq \nu_i(c) \leq q_c$. Moreover, since r' is a progressive run of $\mathcal{R}(N)$, and $\nu_i \in \alpha_i$ for all $i \in I\!N$, then $\exists^\infty_i.\nu_i(c) = 0 \vee \nu_i(c) > q_c$. Since $\exists^\infty_i.\nu_i(c) \leq q_c$, then $\exists^\infty_i.\nu_i(c) = 0$. Since $\exists^\infty_i.\frac{1}{2} \leq \nu_i(c)$, then $\exists^\infty_i.\exists k > i.\tau_k \geq \tau_i + \frac{1}{2}$ and then τ is unbounded.
2. Suppose $\exists^\infty_i.\forall c \in C.\nu_i(c) < \frac{1}{2} \vee q_c < \nu_i(c)$. Then $\exists^\infty_i.\tau_{i+1} \geq \tau_i + \frac{1}{2}$, and then τ is unbounded.

Thus (σ, τ) is a timed sequence and $r \stackrel{\text{def}}{=} t_1 t_2 \ldots$ is a run of N over (σ, τ), with $(m_0, \nu_0), (m_1, \nu_1), \ldots$ as sequence of associated snapshots. Furthermore $[r] = r'$. ■

Finally we define the acceptance family. A set of accepting transitions has to ensure that a run in $\mathcal{R}(N)$ may be accepted only if it is progressive.

Definition 6.15 (Progressive Acceptor) A set $R \subseteq T'$ is a *progressive acceptor* if and only if $\forall c \in C.\exists (t, \alpha', \alpha) \in R.\overline{\alpha}(c) = 0 \vee \overline{\alpha}(c) > q_c$. □

Theorem 6.16 *Given an ATP N, there exists an ABP N' such that $\mathcal{L}(N') = Untime[\mathcal{L}(N)]$.*

Proof: Given $N = (\Sigma, P, T, F, W, h, m_0, C, c^c, c^r, \mathcal{F})$, let $N' \stackrel{\text{def}}{=} (\Sigma, P', T', F', W', h', {m_0}', \mathcal{F}')$ be the ABP where $(\Sigma, P', T', F', W', h', {m_0}')$ is the region net $\mathcal{R}(\Sigma, P, T, F, W, h, m_0, C, c^c, c^r)$ and $\mathcal{F}' \stackrel{\text{def}}{=} \{R' \subseteq T' | \pi_1(R') \in \mathcal{F}$ and R' is a progressive acceptor $\}$. Suppose r' is a run of N' such

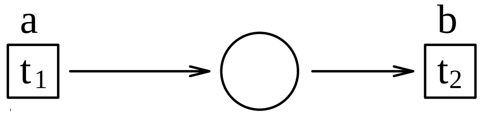

Figure 2: Non complementable net.

that $In(r') \in \mathcal{F}'$. Then $In(r')$ is a progressive acceptor, and then r' must be a progressive run of N'. By Lemma 6.14 there exists a time sequence τ and a run r of N over $(h(r), \tau)$ such that $r' = [r]$. Since T' is finite, $In(r) = \pi_1(In(r'))$; and $In(r) \in \mathcal{F}$ because $In(r') \in \mathcal{F}'$. Thus r is accepted by N and then $h'(r') = h(r) \in Untime[\mathcal{L}(N)]$.
Conversely, assume $r = t_1 t_2 \ldots$ is a run of N on $(h(r), \tau)$ such that $In(r) \in \mathcal{F}$. It is $\pi_1(In([r])) = In(r) \in \mathcal{F}$; by Lemma 6.12, $[r]$ is a progressive run of N', hence $In([r])$ is a progressive acceptor, and then $In([r]) \in \mathcal{F}'$. Thus, $[r]$ is accepted by N', hence $h(r) = h'([r]) \in \mathcal{L}(N')$. ■

Theorem 6.17 *The emptiness problem for ATPs is decidable.*

Proof: $L = \emptyset$ iff $Untime[L] = \emptyset$; hence the thesis follows by Theorem 6.16 and 2.4. ■

7 Expressiveness and Closure Properties

In this section we compare ATPs with timed finite state automata and show some closure properties w.r.t. boolean operations.

Theorem 7.1 *[12, 9]* $\mathcal{TBA} \subset \mathcal{ATP}$.

The basic idea underlying the proof is that every ABTMA can be simulated by an ATP: a place in the net corresponds to a state in the ABTMA. On the contrary, there exist nets recognizing non regular timed languages. A relevant counterexample is illustrated by the net N in Figure 1, with acceptance family $\mathcal{F} \stackrel{\text{def}}{=} \{\{t_1\}, \{t_1, t_2\}\}$. It can be proved that $Untime[\mathcal{L}(N)] = \{\sigma \in \{a, b\}^\omega \mid \forall k \in I\!N. \#\{i \leq k \mid \sigma_i = a\} \geq \#\{i \leq k \mid \sigma_i = b\}\}$ is not a regular ω-language. In [4, 5] Alur and Dill proved that this fact implies that $\mathcal{L}(N)$ is not a timed regular language.

Theorem 7.2 *[12, 9]*

(i) *$\mathcal{ATP}$ is closed under finite union.*

(ii) *$\mathcal{ATP}$ is closed under finite intersection.*

(iii) *$\mathcal{ATP}$ is not closed under complementation.*

To prove that $\mathcal{ATP}$ is not closed under complementation, consider the ATP N in Figure 2 with no clock constraints, and acceptance family $\mathcal{F} \stackrel{\text{def}}{=} \{\{t_1\}, \{t_1, t_2\}\}$. It can be proved that $\mathcal{L}(N) \stackrel{\text{def}}{=} \{(\sigma, \tau) \in \{a, b\}^t \mid \forall k \in I\!N. \#\{i \leq k \mid \sigma_i = a\} \geq \#\{i \leq k \mid \sigma_i = b\}\}$. Thus $\overline{\mathcal{L}(N)} \stackrel{\text{def}}{=} \{(\sigma, \tau) \in \{a, b\}^t \mid \exists k \in I\!N. \#\{i \leq k \mid \sigma_i = a\} < \#\{i \leq k \mid \sigma_i = b\}\}$.
Suppose there exists an ATP N' such that $\mathcal{L}(N') = \overline{\mathcal{L}(N)}$. Then, for Theorem 6.16 there exists an ABP N'' such that $\mathcal{L}(N'') = \{\sigma \in \{a, b\}^\omega \mid \exists k \in I\!N. \#\{i \leq k \mid \sigma_i = a\} < \#\{i \leq k \mid \sigma_i = b\}\}$, but it can be proved that such an ABP does not exist.
We summarize the comparison on the various types of automata in Table 1.

Class of timed languages	Operation closed under
$\mathcal{ATP}$	union, intersection
$\cup$	
$\mathcal{TMA} = \mathcal{TBA} = \mathcal{ABTMA} = \mathcal{ABTBA}$	union, intersection
$\cup$	
$\mathcal{DTMA}$	union, intersection, complementation
$\cup$	
$\mathcal{DTBA}$	union, intersection

Table 1: Classes of timed automata.

8 Verification

In this section we discuss how to use the theory of timed P/T nets to prove correctness of some infinite-state real-time systems. In [3, 4] time is introduced in linear trace semantics [10] for concurrent processes. In the model, a trace will be a timed word of set of events: if two events a and b happen simultaneously, the trace will have the set $\{a, b\}$.

Definition 8.1 (Timed Traces) $\psi \in \wp^+(A)^t$ is a *timed trace* over an alphabet A. □

Definition 8.2 (Timed Process) A *timed process* is a pair (A, L) where A is a finite set of observable events, and L is a timed language over $\wp^+(A)$. The set of timed processes is denoted by $\mathcal{TP}$. The class of processes *modeled* by ATPs is $\mathcal{ATPP} \stackrel{\text{def}}{=} \{(A, L) \in \mathcal{TP} | \, L \in \mathcal{ATP}\}$. □

Remark 8.3 *A property* Π *can be represented as the set of traces satisfying it. Hence verifying that a process* $P = (A, L)$ *satisfies property* Π *is equivalent to check that* $L \subseteq \Pi$*, i.e. that all the execution traces satisfy the property* Π*.*

Various operations can be defined on processes; these are useful for describing complex systems as composed of simpler ones. We will consider only the most important of these operations, namely, parallel composition. The parallel composition of a set of processes prescribes the joint behavior of all the processes running concurrently.
The parallel composition operator can be conveniently defined using the *projection* operation. The projection of $(\sigma, \tau) \in \wp^+(A)^t$ onto $B \subseteq A$ is formed by intersecting each event set in σ with B and deleting all the empty sets from the sequence. Notice that the projection operation may result in a finite sequence, thus we define the set of timed traces *projectable* onto B.

Definition 8.4 (Projectable Timed Traces) $A\lceil B \stackrel{\text{def}}{=} \{(\sigma, \tau) \in \wp^+(A)^t | \, \exists^\infty i. \sigma_i \cap B \neq \emptyset\}$. If $B_1, \ldots, B_n \subseteq A$, we define $A\lceil_i B_i \stackrel{\text{def}}{=} \{\psi \in \wp^+(A)^t | \, \wedge_i \, \psi \in A\lceil B_i\}$. □

Definition 8.5 (Projection) Assume $(\sigma, \tau) \in A\lceil B$. We define $\xi_1 \stackrel{\text{def}}{=} \min\{i \in I\!N^+ | \, \sigma_i \cap B \neq \emptyset\}$, $\xi_{k+1} \stackrel{\text{def}}{=} \min\{i > \xi_k | \, \sigma_i \cap B \neq \emptyset\}$, $\sigma' \stackrel{\text{def}}{=} \sigma_{\xi_1}\sigma_{\xi_2}\ldots$, $\tau' \stackrel{\text{def}}{=} \tau_{\xi_1}\tau_{\xi_2}\ldots$, and $(\sigma, \tau)\lceil B \stackrel{\text{def}}{=} (\sigma', \tau')$. □

Definition 8.6 (Parallel Composition) Assume $P_i = (A_i, L_i)$ is a timed process for $i = 1, 2, \ldots, n$. Their parallel composition is the timed process $\|_i \, P_i \stackrel{\text{def}}{=} (\cup_i A_i, \|_i \, L_i)$ where

$$\|_i \, L_i \stackrel{\text{def}}{=} \{\psi \in (\cup_i A_i)\lceil_j A_j | \, \wedge_j \, \psi\lceil A_j \in L_j\}$$

□

We want to prove that $\mathcal{ATPP}$ is closed under parallel composition. Assume $P_i = (A_i, L_i) \in \mathcal{ATPP}$ for $i = 1, 2, \ldots, n$, and N_i is an ATP such that $\mathcal{L}(N_i) = L_i$. $\mathcal{G}(L_k)$ denotes the timed language $\{\psi \in (\cup_i A_i)\lceil_j A_j \mid \psi\lceil A_k \in L_k\}$. The first step will be, for each $k \in \{1, \ldots, n\}$, to define an ATP $\mathcal{G}(N_k)$ such that $\mathcal{L}(\mathcal{G}(N_k)) = \mathcal{G}(L_k)$.

Definition 8.7 (Generalization Construction)
Given the net $N_k = (\wp^+(A_k), P_k, T_k, F_k, W_k, h_k, m_k, c_k^c, c_k^r, \mathcal{F}_k)$, we define below the net $\mathcal{G}(N_k) \stackrel{\text{def}}{=} (\wp^+(\cup_i A_i), P_k, T_k{}', F_k{}', W_k{}', h_k{}', m_k, c_k^c{}', c_k^r{}', \mathcal{F}_k{}')$ as follows.

- $T_k{}' \stackrel{\text{def}}{=} \overline{A}_k \cup \{(t, A) \mid A = \emptyset \vee A \in \overline{A}_k\}$ where $\overline{A}_k \stackrel{\text{def}}{=} \wp^+(\cup_i A_i \setminus A_k)$; [1]
- $F_k{}' \stackrel{\text{def}}{=} \{(p, (t, A)) \mid (p, t) \in F_k\} \cup \{((t, A), p) \mid (t, p) \in F_k\}$;
- $W_k{}'(p, (t, A)) \stackrel{\text{def}}{=} W_k(p, t)$ and $W_k{}'((t, A), p) \stackrel{\text{def}}{=} W_k(t, p)$;
- $h_k{}'(A) \stackrel{\text{def}}{=} A$ and $h_k{}'(t, A) \stackrel{\text{def}}{=} h_k(t) \cup A$.
- $c_k^c{}'(A) \stackrel{\text{def}}{=} \mathit{true}$ and $c_k^c{}'(t, A) \stackrel{\text{def}}{=} c_k^c(t)$;
- $c_k^r{}'(A) \stackrel{\text{def}}{=} \emptyset$ and $c_k^r{}'(t, A) \stackrel{\text{def}}{=} c_k^r(t)$;
- Given $T \subseteq T_k{}'$, we say that T is *complete* iff $\forall j. \exists t \in T. A_j \cap h_k{}'(t) \neq \emptyset$. When $R \in \mathcal{F}_k$, we say that T *emulates* R iff $\pi_1(T \setminus \overline{A}_k) = R$. Then $\mathcal{F}_k{}' \stackrel{\text{def}}{=} \{T \subseteq T_k{}' \mid T$ is complete and there exists $R \in \mathcal{F}_k$ such that T emulates $R\}$. □

Theorem 8.8 *$\mathcal{ATPP}$ is closed under parallel composition.*

Proof: We want to prove that $\mathcal{L}(\mathcal{G}(N_k)) = \mathcal{G}(L_k)$, and then the thesis follows by Theorem 7.2.(ii). Suppose $\psi \in \mathcal{L}(\mathcal{G}(N_k))$ and r' is a run over ψ accepted by $\mathcal{G}(N_k)$. Since $In(r')$ is complete, then $\psi \in (\cup_i A_i)\lceil_j A_j$. Furthermore there exists $R \in \mathcal{F}_k$ such that $In(r')$ emulates R, hence $\exists_i^\infty . r'_i \notin \overline{A}_k$. Let $(t_{\xi_1}, A_{\xi_1})(t_{\xi_2}, A_{\xi_2}) \ldots$ be the subsequence of these r'_i: we define $r \stackrel{\text{def}}{=} t_{\xi_1} t_{\xi_2} \ldots$. It is easy to prove that r is a run over $\psi\lceil A_k$ accepted by N_k, in fact $In(r) = R$.
Conversely, assume $\psi = (\sigma, \tau) \in \mathcal{G}(L_k)$. Then $\psi\lceil A_k \in L_k$, hence there exists a run r over $\psi\lceil A_k$ accepted by N_k. We define the ω-word $r' \in T_k{}'^\omega$ using an auxiliary sequence ξ, defined as follows: $\xi_1 \stackrel{\text{def}}{=} 1$; if $\sigma_i \cap A_k \neq \emptyset$ then $r'_i \stackrel{\text{def}}{=} (r_{\xi_i}, \sigma_i \setminus h_k(r_{\xi_i}))$ and $\xi_{i+1} \stackrel{\text{def}}{=} \xi_i + 1$, otherwise $r'_i \stackrel{\text{def}}{=} \sigma_i$ and $\xi_{i+1} \stackrel{\text{def}}{=} \xi_i$. It is easy to prove that r' is a run over ψ accepted by $\mathcal{G}(N_k)$. In fact, since $\psi \in (\cup_i A_i)\lceil_j A_j$, $In(r')$ is complete and, furthermore, $In(r')$ emulates $In(r)$. ■

Typically, an implementation of a system is described as a composition of several components. If we assume that every component is a timed process modeled by an ATP and that the specification of the system is given as a property modeled by a DTMA, then the correctness of the implementation can be checked, due to the following

Theorem 8.9 *Given $P_i = (A_i, \mathcal{L}(N_i)) \in \mathcal{ATPP}$, modeled by ATPs N_i (for $i = 1, \ldots, n$), and a specification as a deterministic timed automaton $\mathcal{A}$, the inclusion of $\|_i\ \mathcal{L}(N_i)$ in $\mathcal{L}(\mathcal{A})$ can be decided effectively.*

Proof: By Theorem 8.8, an ATP N that accepts $\|_i\ \mathcal{L}(N_i)$ can be built. Furthermore $\mathcal{L}(N) \subseteq \mathcal{L}(\mathcal{A})$ iff $\mathcal{L}(N) \cap \overline{\mathcal{L}(\mathcal{A})} \neq \emptyset$. By Theorem 4.6, a DTMA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \overline{\mathcal{L}(\mathcal{A})}$ can be built.

[1] If $A \in \overline{A}_k$, then events in A can happen at any time, thus we introduce a new transition for each of such an A; furthermore, events in A can happen contemporaneously to events in A_k hence, for each transition $t \in T_k$, we introduce a new transition (t, A).

By Theorem 7.1, an ATP N' such that $\mathcal{L}(N') = \mathcal{L}(\mathcal{A}')$ can also be constructed. By Theorem 7.2, an ATP N'' such that $\mathcal{L}(N'') = \mathcal{L}(N) \cap \mathcal{L}(N')$ can be defined, too. Thus $\mathcal{L}(N'') = \mathcal{L}(N) \cap \overline{\mathcal{L}(\mathcal{A})}$, and then $\mathcal{L}(N) \subseteq \mathcal{L}(\mathcal{A})$ iff $\mathcal{L}(N'') = \emptyset$. Hence the thesis follows by Theorem 6.17. ■

9 A Verification Example

An example of automatic verification using ATPs is presented. We model a simple process for producing wooden black horses. In Figure 3 we show an ATP that models a workman that, received a raw block of wood from the environment, carves it producing a wooden horse. We impose that both transitions must be fired infinitely often: thus the process ensures that whenever a new block is provided, then a new horse is refined within no more than 2 minutes. Notice that the workman does not accept a new block until it has completely carved the previous one. This process could have been modeled by a timed finite state automaton, too.

In Figure 4 we show an ATP modeling a workman that obtains a wooden horse and paints it. Again, we impose that both transitions must be repeated infinitely often. The painter can accept more than one horse (we could say that he can put them on a shelf, and this operation makes him busy). However, if no new horse arrives for at least 5 minutes, then the workman has the time to concentrate on painting, and a black horse is produced. Since the process 'remembers' the number of carved horses that are still unpainted, this process cannot be modeled by a timed finite state automaton.

The complete process for the production of black horses is HORSE-MANUFACTORY $\stackrel{\text{def}}{=}$ CARVER || PAINTER. The factory administrator could be tempted to require that if a new block of wood is provided then, in no more than 7 minutes, a new black horse is produced. This property is modeled by the DTMA in Figure 5 (b denotes "block of wood", and h denotes "black horse"): an edge label b stands for any event set containing b, and an edge label ¬b means any event set not containing b. We impose that both states are reached infinitely often in an accepted run.

Using Theorem 8.9, it can be proved that HORSE-MANUFACTORY does not satisfy this property. On the other hand, the administrator could be pleased with the HORSE-MANUFACTORY ensuring that if a new block is provided and for at least 7 minutes no new block is supplied, allowing all workmen to concentrate on the refining job, then at least a black horse is completed. This property is modeled by the DTMA in Figure 6, in which a run is accepted only if it reaches both states infinitely often. Using Theorem 8.9, it can be proved that HORSE-MANUFACTORY satisfies this weaker property.

References

[1] M. Abadi, L. Lamport. "An Old-Fashioned Recipe for Real Time". In *Proceedings of the REX workshop "Real-Time: Theory in Practice"*, LNCS 600, Springer-Verlag, 1992.

[2] R. Alur. "Techniques for Automatic Verification of Real-Time Systems". PhD thesis, Stanford University, 1991.

[3] R. Alur, D. Dill. "Automata for Modeling Real-Time Systems". In *ICALP 90:Automata, Languages, and Programming*, LNCS 443, 322-335. Springer-Verlag, 1990

[4] R. Alur, D. Dill. "The Theory of Timed Automata". In *Proceedings of the REX workshop "Real-Time: Theory in Practice"*, LNCS 600, 45-73. Springer-Verlag, 1992.

[5] R. Alur, D. Dill. "A Theory of Timed Automata". To appear in Theoretical Computer Science.

[6] H. Carstensen, R. Valk. "Infinite Behavior and Fairness in Petri Nets". *Advances in Petri Nets 1984*, LNCS 188, 83-100. Springer-Verlag, 1985.

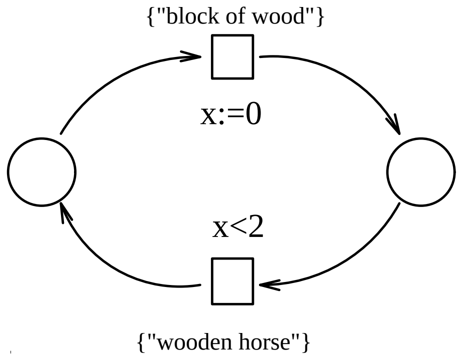

Figure 3: CARVER.

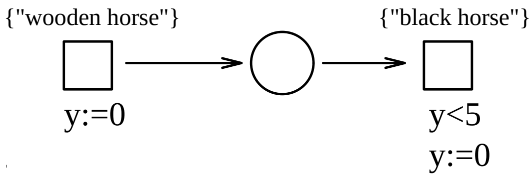

Figure 4: PAINTER.

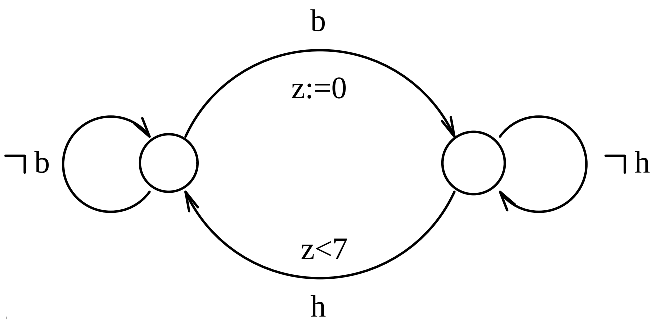

Figure 5: Strong Efficiency Property.

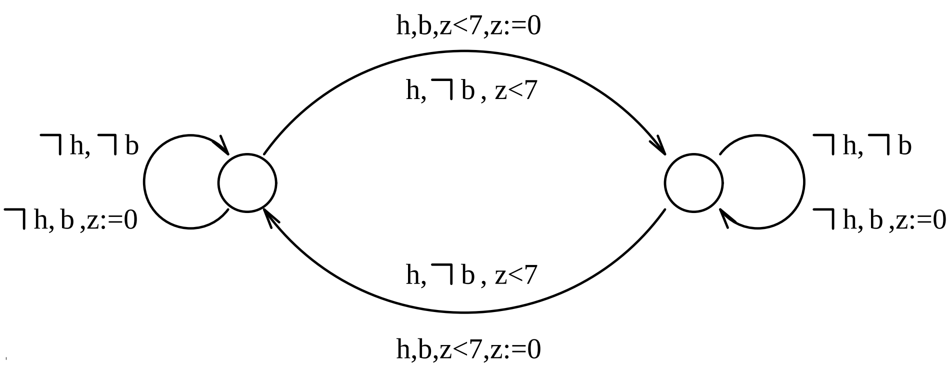

Figure 6: Weak Efficiency Property.

[7] R. Büchi. "On a Decision Method in Restricted Second-Order Arithmetic". In *Proc. of the Int. Con. on Logic, Meth. and Phil. of Science 1960*, Stanford University Press, 1-12, 1962.

[8] D. Dill. "Timing Assumptions and Verification of Finite-State Concurrent Systems". In *Automatic Verification Methods for Finite State Systems*, LNCS 407. Springer-Verlag, 1989.

[9] R. Gorrieri, G. Siliprandi. "A Theory of P/T Nets for Timed Languages". In preparation.

[10] C. Hoare. "Communicating Sequential Processes". Communications of the ACM 21(8), 666-677, 1978.

[11] W. Reisig. "Petri Nets: An Introduction". Springer-Verlag, 1985.

[12] G. Siliprandi. "Metodi Formali per la Descrizione di Processi Temporizzati". Master thesis, Università di Bologna, 1993.

[13] W. Thomas. "Automata on Infinite Objects". Handbook of Theoretical Computer Science. Elsevier Science Publishers, 133-191, 1990.

[14] R. Valk. "Infinite Behavior of Petri Nets". Theoretical Computer Science 25, 311-341, 1983.

[15] R. Valk, M. Jantzen. "The Residue of Vector Sets with Applications to Decidability Problems in Petri Nets". Acta Informatica 21, 643-647, 1985.