# The Power of Reachability Testing for Timed Automata

Luca ACETO[1] *, Patricia BOUYER[2], Augusto BURGUEÑO[3] ** and Kim G. LARSEN[1]

[1] BRICS†, Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7-E, DK-9220 Aalborg Ø, Denmark.
Email: {luca,kgl}@cs.auc.dk, Fax: +45 98 15 98 89
[2] Laboratoire Spécification et Vérification, CNRS UMR 8643, Ecole Normale Supérieure de Cachan,
61, avenue du Président Wilson, F-94235 Cachan Cedex, France.
Email: bouyer@lsv.ens-cachan.fr, Fax: +33 1 47 40 24 64
[3] European Commission, Research Directorate-General
Rue de la Loi 200, B-1049 Brussels, Belgium.
Email: Augusto.Burgueno-Arjona@cec.eu.int

**Abstract.** The computational engine of the verification tool UPPAAL consists of a collection of efficient algorithms for the analysis of reachability properties of systems. Model-checking of properties other than plain reachability ones may currently be carried out in such a tool as follows. Given a property $\phi$ to model-check, the user must provide a test automaton $T_\phi$ for it. This test automaton must be such that the original system $S$ has the property expressed by $\phi$ precisely when none of the distinguished reject states of $T_\phi$ can be reached in the synchronized parallel composition of $S$ with $T_\phi$. This raises the question of which properties may be analyzed by UPPAAL in such a way. This paper gives an answer to this question by providing a complete characterization of the class of properties for which model-checking can be reduced to reachability testing in the sense outlined above. This result is obtained as a corollary of a stronger statement pertaining to the compositionality of the property language considered in this study. In particular, it is shown that our language is the least expressive compositional language that can express a simple safety property stating that no reject state can ever be reached.

Finally, the property language characterizing the power of reachability testing is used to provide a definition of characteristic properties with respect to a timed version of the ready simulation preorder, for nodes of $\tau$-free, deterministic timed automata.

## 1 Introduction

Model-checking of real time systems has been extensively studied in the last ten years, leading to both important theoretical results, setting the limits of decidability [AD94, HK94, HKPV95, BDFP00, Mar00], and tools as HYTECH [HHWT95, HHWT97], KRONOS [OY93, Yov97, BDM+98] and UPPAAL [BLL+95, BLL+98, ABB+01], which have been successfully applied to verification of real sized systems [BGK+96, DY95, HWT95, BF99, BS00, IKL+00]. The main motivation for the work presented in this paper stems from our practical experience with the verification tool UPPAAL, a tool for verifying behavioural properties of real–time systems specified as networks of timed automata [AD94].

One of the main design criteria behind UPPAAL has been that of efficiency. Consequently the computational engine of UPPAAL has originally been restricted to a collection of efficient algorithms for analyzing simple *reachability* properties of a system. However, in practice one often wants to examine a developed model for a number of properties other than what can be directly expressed via reachability. The way model-checking properties other than plain reachability ones

* Partially supported by the Human Capital and Mobility project EXPRESS, a grant from the Italian CNR, Gruppo Nazionale per l'Informatica Matematica (GNIM), and a visiting professorship at the Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan.

** Partially supported by a Research Grant of the Spanish Ministry of Education and Culture and by BRICS. This work was partially carried out while the author was visiting Aalborg University.

† Basic Research in Computer Science.

may currently be carried out in UPPAAL is as follows. Given a property $\phi$ to model-check, the user must provide a *test automaton* $T_\phi$ for the property. The test automaton must be such that the original system $S$ has the property expressed by $\phi$ precisely when none of the distinguished reject states of $T_\phi$ can be reached in the composite system that results when the test automaton is made to interact with the system under investigation. (In the remainder of this introduction, we shall write $S \parallel T_\phi$ for the composite system resulting from the parallel composition of $S$ and the test automaton $T_\phi$.) Intuitively, the test automaton $T_\phi$ monitors the behaviour of the system $S$ looking for violations of the property $\phi$.

Clearly, this raises the question as to which properties may be analyzed by UPPAAL in this manner. In this paper we offer the answer to this question by providing a complete characterization of the class of properties of (networks of) timed automata for which model-checking can be reduced to reachability testing in the above sense.

This paper collects, and extends, all the results of the extended abstracts [ABL98, ABBL98]. In our study, we first consider the property language SBLL suitable for expressing safety and bounded liveness properties of real-time systems. In particular, we show that SBLL is *testable*, in the sense that suitable test automata may be derived for any property of SBLL. However, SBLL is not *expressive complete* with respect to testing, in the sense that it does not embody all properties for which test automata can be derived. We thus present an extension, called $L_{\forall S}$, of SBLL which we prove characterizes the precise limit of the testing approach. More precisely we show that:

- any property $\psi$ of $L_{\forall S}$ is testable, in the sense that there exists a test automaton $T_\psi$ such that any timed automaton $S$ satisfies $\psi$ if and only if $S \parallel T_\psi$ cannot reach a reject state; and
- any test automaton $T$ is expressible in $L_{\forall S}$, in the sense that there exists a formula $\psi_T$ of $L_{\forall S}$ such that, for every timed automaton $S$, the composite system $S \parallel T$ cannot reach a reject state if and only if $S$ satisfies $\psi_T$.

Now let $L_{bad}$ be the property language with only one single property $\phi_{nb}$, expressing that no reject state can ever be reached (a simple safety property). Then the above expressive completeness result will be obtained as an easy corollary of the following stronger compositionality result:

*$L_{\forall S}$ is the least expressive, compositional extension of $L_{bad}$.*

We say that a property language is *compositional* if properties $\phi$ of a composite system $S \parallel T$ can be reduced to sufficient and necessary properties $\phi_T$ of the component $S$. As $\phi_T$ is required to be expressible in the property language, this clearly puts a demand on its expressiveness.

In fact, the above results will not hold for our original and natural property language SBLL, as we will prove that the extension $L_{\forall S}$ has a strictly larger expressive power. In particular we shall demonstrate that there are properties of $L_{\forall S}$ which are not expressible in SBLL.

Having identified the property language $L_{\forall S}$ characterizing the limit of testability, we offer a definition of *characteristic properties* [SI94] with respect to a timed version of the ready simulation preorder (also known as $\frac{2}{3}$-bisimulation) [LS91, BIM95], for nodes of $\tau$-free, deterministic timed automata. This allows one to compute behavioural relations via our model-checking technique, thus effectively reducing the problem of checking the existence of a behavioural relation among states of a timed automaton to a reachability problem. As the version of the ready simulation preorder we consider preserves the properties expressible in $L_{\forall S}$, our model-checking technique may be used to formally and automatically justify abstraction steps in hierarchical system verifications.

The paper is organized as follows. After reviewing the variation on the model of timed automata which is considered in this study (Section 2), we introduce test automata and describe how they can be used to test for properties *via* reachability analysis (see Section 3). The property languages studied in this paper, namely SBLL and $L_{\forall S}$, are presented and proved testable in Section 4. In Section 5, we prove that the property language $L_{\forall S}$ is expressive complete with respect to reachability testing. In Section 6, we provide a behavioural characterization of the preorder on states induced by the property language $L_{\forall S}$ over a subclass of timed automata. A key step in such a characterization is to show how the property language $L_{\forall S}$ can be used to define characteristic properties for nodes of $\tau$-free, deterministic timed automata with respect to a timed version of the ready simulation preorder. The paper concludes with some final remarks (Section 7). The technical appendices contain proofs of results that are omitted from the main body of the paper for the sake of readability.

## 2 Preliminaries

We begin by briefly reviewing a variation on the timed automaton model proposed by Alur and Dill [AD94] and the time labelled transition systems that underlie its semantics.

### 2.1 Timed Labelled Transition Systems

Let $\mathcal{A}$ be a finite set of *actions*, and $\mathcal{U}$ be a finite set of *urgent actions* disjoint from $\mathcal{A}$. We use Act to stand for $\mathcal{A} \cup \mathcal{U}$ and let $a, b, c$ range over it. We assume that Act comes equipped with a mapping $\overline{\cdot} : \mathrm{Act} \to \mathrm{Act}$ such that, for every $a \in \mathrm{Act}$, it holds that $\overline{\overline{a}} = a$. Moreover, we require that $\overline{a} \in \mathcal{A}$ iff $a \in \mathcal{A}$, for every action $a$. (Note that, since $\mathcal{A}$ and $\mathcal{U}$ are disjoint, it is also the case that $\overline{a} \in \mathcal{U}$ iff $a \in \mathcal{U}$.) We let $\mathrm{Act}_\tau$ stand for $\mathrm{Act} \cup \{\tau\}$, where $\tau$ is a symbol not occurring in Act, and use $\mu$ to range over it. Following Milner [Mil89], $\tau$ will stand for an internal action of a system. Let $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ denote the sets of natural and non-negative real numbers, respectively. We use $\mathcal{D}$ to denote the set of *delay actions* $\{\epsilon(d) \mid d \in \mathbb{R}_{\geq 0}\}$, and $\mathcal{L}$ to stand for the union of $\mathrm{Act}_\tau$ and $\mathcal{D}$. The meta-variable $\alpha$ will range over $\mathcal{L}$.

**Definition 2.1.** A *timed labelled transition system* (TLTS) is a structure $\mathcal{T} = \langle \mathcal{S}, \mathcal{L}, s^0, \longrightarrow \rangle$ where $\mathcal{S}$ is a set of *states*, $s^0 \in \mathcal{S}$ is the initial state, and $\longrightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is a transition relation satisfying the following properties:

- TIME DETERMINISM: for every $s, s', s'' \in \mathcal{S}$ and $d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\epsilon(d)} s'$ and $s \xrightarrow{\epsilon(d)} s''$, then $s' = s''$;
- TIME ADDITIVITY: for every $s, s'' \in \mathcal{S}$ and $d_1, d_2 \in \mathbb{R}_{\geq 0}$, $s \xrightarrow{\epsilon(d_1+d_2)} s''$ iff $s \xrightarrow{\epsilon(d_1)} s' \xrightarrow{\epsilon(d_2)} s''$, for some $s' \in \mathcal{S}$;
- 0-DELAY: for every $s, s' \in \mathcal{S}$, $s \xrightarrow{\epsilon(0)} s'$ iff $s = s'$;
- FORWARD PERSISTENCE OF URGENT ACTIONS: for every $s, s', s'' \in \mathcal{S}$, $a \in \mathcal{U}$ and $d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\epsilon(d)} s'$ and $s \xrightarrow{a} s''$, then there exists $\overline{s} \in \mathcal{S}$ such that $s' \xrightarrow{a} \overline{s}$;
- BACKWARD PERSISTENCE OF URGENT ACTIONS: for every $s, s', s'' \in \mathcal{S}$, $a \in \mathcal{U}$ and $d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\epsilon(d)} s'$ and $s' \xrightarrow{a} s''$, then there exists $\overline{s} \in \mathcal{S}$ such that $s \xrightarrow{a} \overline{s}$.

As usual, we write $s \xrightarrow{\alpha}$ to mean that there is some state $s'$ such that $s \xrightarrow{\alpha} s'$, and $s \not\xrightarrow{\alpha}$ if there is no state $s'$ such that $s \xrightarrow{\alpha} s'$.

The axioms of time determinism, time additivity and 0-delay are standard in the literature on TCCS (see, *e.g.*, [Yi90]). Those dealing with urgent actions are motivated by the particular kind of timed automaton model considered in a verification tool like UPPAAL. (see Section 2.2 below for details.)

A *delaying computation* is a possibly empty sequence of transitions $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} s_n$ ($n \geq 0$) such that $\alpha_i = \tau$ or $\alpha_i \in \mathcal{D}$, for every $i \in \{1, .., n\}$.

For every state $s$ and delay $d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{\epsilon(d)}$ then we use $\mathrm{delay}(s, d)$ to denote the unique state reachable from $s$ via a $\xrightarrow{\epsilon(d)}$ transition. Following [Yi90], we now proceed to define versions of the transition relations that abstract away from the internal evolution of states as follows:

$$\begin{array}{lll} s \overset{a}{\Longrightarrow} s' & \text{iff} & \exists s''.\ s \xrightarrow{\tau}^{*} s'' \xrightarrow{a} s' \\ s \overset{\epsilon(d)}{\Longrightarrow} s' & \text{iff} & \text{there exists a delaying computation} \\ & & \quad s = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} s_n = s' \quad (n \geq 0) \text{ where} \\ & & \quad d = \sum \{d_i \mid \alpha_i = \epsilon(d_i)\} \ . \end{array}$$

By convention, if the set $\{d_i \mid \alpha_i = \epsilon(d_i)\}$ is empty, then $\sum\{d_i \mid \alpha_i = \epsilon(d_i)\}$ is $0$. With this convention, the relation $\stackrel{\epsilon(0)}{\Longrightarrow}$ coincides with $\stackrel{\tau}{\longrightarrow}^*$, *i.e.*, the reflexive, transitive closure of $\stackrel{\tau}{\longrightarrow}$. Note that the derived transition relation $\stackrel{a}{\Longrightarrow}$ only abstracts from internal transitions *before* the actual execution of action $a$.

**Definition 2.2.** Let $\gamma = s_0 \stackrel{\alpha_1}{\longrightarrow} s_1 \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_n}{\longrightarrow} s_n$ $(n \geq 0)$ be a delaying computation, and let $d = \sum\{d_i \mid \alpha_i = \epsilon(d_i)\}$. For all $d' \in [0, d]$, the *first* state in the computation $\gamma$ reachable from $s_0$ by delaying $d'$ units of time, notation fstate$(\gamma, d')$, is defined thus:

$$\begin{array}{ll}
\mathrm{fstate}(\gamma, d') \stackrel{\mathrm{def}}{=} \mathrm{delay}(s_0, d') & \text{if } \alpha_1 = \epsilon(d_1) \text{ and } 0 \leq d' < d_1, \text{ or } d' = 0 \\
\mathrm{fstate}(\gamma, d') \stackrel{\mathrm{def}}{=} \mathrm{fstate}(s_1 \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_n}{\longrightarrow} s_n, d') & \text{if } \alpha_1 = \tau \text{ and } d' > 0 \\
\mathrm{fstate}(\gamma, d') \stackrel{\mathrm{def}}{=} \mathrm{fstate}(s_1 \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_n}{\longrightarrow} s_n, d' - d_1) & \text{if } \alpha_1 = \epsilon(d_1) \text{ and } d' \geq d_1 > 0 \enspace .
\end{array}$$

We define a collection of transition relations parameterized by a set of *urgent* actions $S$ as follows:

$$\begin{array}{lll}
s \stackrel{\mu}{\longrightarrow}_S s' & \text{iff} & s \stackrel{\mu}{\longrightarrow} s' \\
s \stackrel{\epsilon(d)}{\longrightarrow}_S s' & \text{iff} & s \stackrel{\epsilon(d)}{\longrightarrow} s' \text{ and } \forall d' \in [0, d[,\ a \in S.\ \mathrm{delay}(s, d') \stackrel{a}{\not\longrightarrow} \\
s \stackrel{a}{\Longrightarrow}_S s' & \text{iff} & s \stackrel{a}{\Longrightarrow} s' \\
s \stackrel{\epsilon(d)}{\Longrightarrow}_S s' & \text{iff} & \text{there exists a delaying computation} \\
 & & s = s_0 \stackrel{\alpha_1}{\longrightarrow}_S s_1 \stackrel{\alpha_2}{\longrightarrow}_S \ldots \stackrel{\alpha_n}{\longrightarrow}_S s_n = s' \quad (n \geq 0) \text{ where} \\
 & & d = \sum\{d_i \mid \alpha_i = \epsilon(d_i)\} \enspace .
\end{array}$$

Intuitively, $s \stackrel{\epsilon(d)}{\longrightarrow}_S s'$ holds if $s$ can delay $d$ units of time, and no action in the set $S$ becomes enabled before time $d$ during this delay activity. Note that, since the set $S$ only contains urgent actions, this amounts to requiring that either $d = 0$ or $s \stackrel{a}{\not\longrightarrow}$, for every $a \in S$. Similarly, $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$ holds if there exists a delaying computation of duration $d$ from state $s$ whose delay transitions with positive duration occur only in states in which none of the urgent actions in $S$ are enabled.

*Example 2.3.* Consider the TLTS whose states are $s$ and $s'$. The action transitions from these states are $s \stackrel{a}{\longrightarrow} s$, where $a$ is urgent, and $s \stackrel{\tau}{\longrightarrow} s'$. Their delay transitions are $s \stackrel{\epsilon(d)}{\longrightarrow} s$ and $s' \stackrel{\epsilon(d)}{\longrightarrow} s'$ for every $d \in \mathbb{R}_{\geq 0}$. Then, for every $d \in \mathbb{R}_{\geq 0}$, state $s$ affords the following delaying computation: $s \stackrel{\tau}{\longrightarrow} s' \stackrel{\epsilon(d)}{\longrightarrow}_{\{a\}} s'$. Thus $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ for every $d \in \mathbb{R}_{\geq 0}$.

Note that, as in the above example, a delaying computation justifying the transition $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$ *can* traverse states where actions in $S$ are enabled, but no actual delaying is permitted in them.

The notion of fstate$(s_0 \stackrel{\alpha_1}{\longrightarrow}_S s_1 \stackrel{\alpha_2}{\longrightarrow}_S \ldots \stackrel{\alpha_n}{\longrightarrow}_S s_n, d')$ can be defined exactly as in Definition 2.2. In what follows, this notion will also be applied to transitions of the form $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$. In that case, we shall always be implicitly considering an arbitrary computation that justifies the $\tau$-abstracting transition under consideration.

**Definition 2.4.** Let $\mathcal{T} = \langle \mathcal{S}, \mathcal{L}, s^0, \rightarrow \rangle$ be a TLTS and let $L \subseteq \mathsf{Act}$ be a set of actions. The restriction of $\mathcal{T}$ over $L$ is the TLTS $\mathcal{T} \backslash L = \langle \mathcal{S} \backslash L, \mathcal{L}, s^0 \backslash L, \leadsto \rangle$, where $\mathcal{S} \backslash L = \{s \backslash L \mid s \in \mathcal{S}\}$ and the transition relation $\leadsto$ is defined by the rules in Table 2.

**Definition 2.5.** Let $\mathcal{T}_i = \langle \mathcal{S}_i, \mathcal{L}, s_i^0, \longrightarrow_i \rangle$ $(i \in \{1, 2\})$ be two TLTSs. The parallel composition of $\mathcal{T}_1$ and $\mathcal{T}_2$ is the TLTS

$$\mathcal{T}_1 \parallel \mathcal{T}_2 = \langle \mathcal{S}_1 \times \mathcal{S}_2, \mathcal{L}, (s_1^0, s_2^0), \longrightarrow \rangle$$

where the transition relation $\longrightarrow$ is defined by the rules in Table 1. *Ibidem*, and in the remainder of the paper, we use the more suggestive notation $s \parallel s'$ in lieu of $(s, s')$.

| | | | |
|---|---|---|---|
| (1) | $\dfrac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \| s_2 \xrightarrow{\mu} s_1' \| s_2}$ | (2) | $\dfrac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \| s_2 \xrightarrow{\mu} s_1 \| s_2'}$ |
| (3) | $\dfrac{s_1 \xrightarrow{a}_1 s_1' \qquad s_2 \xrightarrow{\overline{a}}_2 s_2'}{s_1 \| s_2 \xrightarrow{\tau} s_1' \| s_2'}$ | | |
| (4) | $\dfrac{s_1 \xrightarrow{\epsilon(d)}_1 s_1' \qquad s_2 \xrightarrow{\epsilon(d)}_2 s_2'}{s_1 \| s_2 \xrightarrow{\epsilon(d)} s_1' \| s_2'}$ | | $d = 0$ or $\forall a \in \mathcal{U}.$ $\neg(s_1 \xrightarrow{a}_1 \ \wedge \ s_2 \xrightarrow{\overline{a}}_2)$ |

where $s_i, s_i'$ are states of $\mathcal{T}_i$ $(i \in \{1, 2\})$,
$\mu \in \mathsf{Act}_\tau$, $a, \overline{a} \in \mathsf{Act}$ and $d \in \mathbb{R}_{\geq 0}$.

**Table 1.** Rules defining the transition relation $\rightarrow$ in $\mathcal{T}_1 \parallel \mathcal{T}_2$

| | | | |
|---|---|---|---|
| (1) | $\dfrac{s \xrightarrow{\tau} s'}{s \backslash L \overset{\tau}{\rightsquigarrow} s' \backslash L}$ | (2) | $\dfrac{s \xrightarrow{\epsilon(d)} s'}{s \backslash L \overset{\epsilon(d)}{\rightsquigarrow} s' \backslash L}$ |
| (3) | $\dfrac{s \xrightarrow{a} s'}{s \backslash L \overset{a}{\rightsquigarrow} s' \backslash L}$ | | $a, \overline{a} \notin L$ |

where $s, s'$ are states of $\mathcal{T}$,
$L \subseteq \mathsf{Act}$, $a \in \mathsf{Act}$, and $d \in \mathbb{R}_{\geq 0}$.

**Table 2.** Rules defining the transition relation $\rightsquigarrow$ in $\mathcal{T} \backslash L$

The reader familiar with TCCS [Yi90] may have noticed that the above definition of parallel composition has strong similarities with that of TCCS parallel composition — the only difference being that in TCCS *all* actions are urgent. The definition of $\|$ forces the composed TLTSs to synchronize on delays, with the particularity that delaying by a positive amount of time is only possible when no synchronization on urgent actions is. (see the side condition of rule (4) in Table 1.) This yields precisely the parallel composition operator used in UPPAAL [BLL+95].

The reader will easily realize that the following holds.

**Proposition 2.6.** *The class of TLTSs is closed under the operations of parallel composition and restriction.*

**Definition 2.7.** Let $\mathcal{T}_i = \langle \mathcal{S}_i, \mathcal{L}, s_i^0, \longrightarrow_i \rangle$ $(i \in \{1, 2\})$ be two TLTSs. We say that $\mathcal{T}_1$ and $\mathcal{T}_2$ are *isomorphic* iff there exists a bijection $h : \mathcal{S}_1 \rightarrow \mathcal{S}_2$ such that

1. $h(s_1^0) = s_2^0$ and
2. $s \xrightarrow{\alpha}_1 s'$ iff $h(s) \xrightarrow{\alpha}_2 h(s')$, for every $s, s' \in \mathcal{S}_1$ and $\alpha \in \mathcal{L}$.

### 2.2 Timed Automata

Let $C$ be a set of clocks. We use $\mathcal{B}(C)$ to denote the set of boolean expressions over atomic formulae of the form $x \sim p$ and $x - y \sim p$, with $x, y \in C$, $p \in \mathbb{N}$, and $\sim \in \{<, >, =\}$. Expressions in $\mathcal{B}(C)$ are interpreted over the collection of time assignments. A *time assignment*, or *valuation*, $v$ for $C$ is a function from $C$ to $\mathbb{R}_{\geq 0}$. Given a condition $g \in \mathcal{B}(C)$ and a time assignment $v$, the boolean value $g(v)$ describes whether $g$ is satisfied by $v$ or not. (Note that $\mathcal{B}(C)$ is closed under negation.) For every time assignment $v$ and $d \in \mathbb{R}_{\geq 0}$, we use $v + d$ to denote the time assignment which maps each clock $x \in C$ to the value $v(x) + d$. Two assignments $u$ and $v$ are said to agree on the set of clocks $C'$ iff they assign the same real number to every clock in $C'$. For every subset $C'$ of clocks, $[C' \to 0]v$ denotes the assignment for $C$ which maps each clock in $C'$ to the value $0$ and agrees with $v$ over $C \backslash C'$. For an assignment $u$ and a subset $C'$ of $C$, we write $u \upharpoonright C'$ for the restriction of $u$ to the set of clocks $C'$. Given two disjoint sets of clocks $C_1$, $C_2$, and two valuations $v_1$, $v_2$ for the clocks of $C_1$ and $C_2$ respectively, $v_1 : v_2$ denotes the valuation for the clocks of $C_1 \cup C_2$ such that $(v_1 : v_2)(x) = v_1(x)$ iff $x \in C_1$ and $(v_1 : v_2)(x) = v_2(x)$ iff $x \in C_2$.

The notion of timed automaton we use in this paper is a variation on the original one introduced by Alur and Dill [AD94], and underlies that used in, *e.g.*, UPPAAL [BLL+95] and KRONOS [DY95].

**Definition 2.8.** A *timed automaton* is a tuple $A = \langle \mathsf{Act}_\tau, N, n_0, C, E \rangle$ where $N$ is a finite set of *nodes*, $n_0$ is the *initial node*, $C$ is a finite set of *clocks*, and $E \subseteq N \times N \times \mathsf{Act}_\tau \times 2^C \times \mathcal{B}(C)$ is a set of *edges*. The tuple $e = \langle n, n_e, \mu, r_e, g_e \rangle \in E$ stands for an edge from node $n$ to node $n_e$ (the *target* of $e$) with action $\mu$, where $r_e$ denotes the set of clocks to be reset to $0$ and $g_e$ is the enabling condition (or *guard*) over the clocks of $A$. All the timed automata we shall consider in this paper will satisfy the following constraint:

- URGENCY: if $\langle n, n_e, \mu, r_e, g_e \rangle \in E$ and $\mu \in \mathcal{U}$, then $g_e$ is a tautology, *i.e.*, $g_e$ is satisfied by every valuation for the clocks in $C$.

For every node $n$ and action $\mu$, we use $E(n, \mu)$ to denote the set of edges emanating from $n$ whose action is $\mu$. The collection of nodes of a timed automaton $A$ will be written $\textsc{Nodes}(A)$. For every node $n$ we define $\mathsf{Act}(n)$ to be the set of actions in $\mathsf{Act}$ labelling all edges emanating from it, *i.e.* $\mathsf{Act}(n) = \{a \in \mathsf{Act} \mid E(n, a) \neq \emptyset\}$, and use $\mathcal{U}(n)$ to stand for the collection of urgent actions contained in $\mathsf{Act}(n)$.

*Remark 2.9.* The difference between the notion of timed automaton presented above and the classic one by Alur and Dill lies in the presence of urgent actions. These have been introduced in the models underlying the verification tools KRONOS and UPPAAL to model composite systems where progress (interaction) is guaranteed to happen within certain upper time bounds. Urgency of actions actually predates timed automata and goes back to, *e.g.*, the original calculus of Timed CCS (TCCS) by Wang Yi [Yi90, Yi91].

**Convention.** In what follows, we shall assume that the clocks used in timed automata come from a fixed, countably infinite collection of clocks $C_A$.

*Example 2.10.* The timed automaton depicted in Figure 1 has five nodes labelled $n_0$ to $n_4$, one clock $x$, actions $a \in \mathcal{U}$ and $b \in \mathcal{A}$, and four edges. The edge from node $n_1$ to node $n_2$, for example, is guarded by $x \geq 0$, is labelled with the urgent action $a$ and resets clock $x$. Note that the guards of edges labelled with the urgent action $a$ are tautologies.

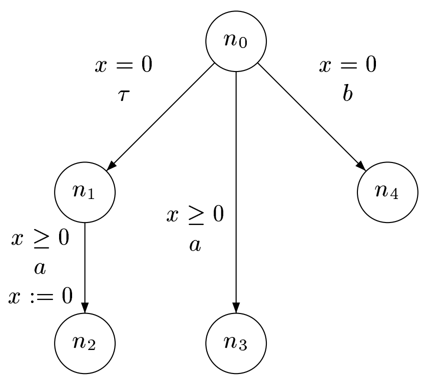



**Figure 1.** Timed automaton $A$ ($a \in \mathcal{U}$ and $b \in \mathcal{A}$)

A *state* of a timed automaton $A$ is a pair $\langle n, v\rangle$ where $n$ is a node of $A$ and $v$ is a time assignment for $C$. The initial state of $A$ is $\langle n_0, v_0\rangle$ where $n_0$ is the initial node of $A$ and $v_0$ is the time assignment mapping all clocks in $C$ to $0$.

The operational semantics of a timed automaton $A$ is given by the TLTS $\mathcal{T}_A = \langle \mathcal{S}, \mathcal{L}, \sigma^0, \longrightarrow\rangle$, where $\mathcal{S}$ is the set of states of $A$, $\sigma^0$ is the initial state of $A$, and $\longrightarrow$ is the transition relation defined as follows:

$$\begin{array}{lll}\langle n, v\rangle \xrightarrow{\mu} \langle n', v'\rangle & \text{iff} & \exists e = \langle n, n', \mu, r_e, g_e\rangle \in E.\ g_e(v) \wedge v' = [r_e \to 0]v \\ \langle n, v\rangle \xrightarrow{\epsilon(d)} \langle n', v'\rangle & \text{iff} & n = n' \text{ and } v' = v + d\end{array}$$

where $\mu \in \mathsf{Act}_\tau$ and $\epsilon(d) \in \mathcal{D}$.

*Example 2.11.* The following is a valid sequence of transitions for the timed automaton of Figure 1, where the number in brackets corresponds to the time assignment of clock $x$:

$$\langle n_0, \{0\}\rangle \xrightarrow{\tau} \langle n_1, \{0\}\rangle \xrightarrow{\epsilon(3.14)} \langle n_1, \{3.14\}\rangle \xrightarrow{a} \langle n_2, \{0\}\rangle\ .$$

**Proposition 2.12.** *Let $A$ be a timed automaton. Then $\mathcal{T}_A$ is a TLTS.*

**Definition 2.13.** Let $A = \langle \mathsf{Act}_\tau, N_A, n_A, C_A, E_A\rangle$ and $B = \langle \mathsf{Act}_\tau, N_B, n_B, C_B, E_B\rangle$ be timed automata. We say that $A$ and $B$ are *isomorphic* iff there exists a pair $h = \langle h_N, h_C\rangle$ of bijections such that:

1. $h_N : N_A \to N_B$,
2. $h_C : C_A \to C_B$,
3. $h_N(n_A) = n_B$, and
4. $\langle n, n', \mu, r, g\rangle \in E_A$ iff $\langle h_N(n), h_N(n'), \mu, h_C(r), h_C(g)\rangle \in E_B$, where $h_C(g)$ denotes the boolean condition obtained by substituting every occurrence of a clock $x$ in $g$ with the clock $h_C(x)$.

It is easy to see that isomorphic timed automata yield isomorphic TLTSs.

## 3 Testing Automata

As mentioned in Section 1, the main aim of this paper is to present a complete characterization of the class of properties of (networks of) timed automata for which model-checking can be reduced to reachability analysis. In this section we take the first steps towards the definition of *model-checking via (reachability) testing* by defining *testing* [DNH84]. Informally, testing involves the parallel composition of the tested automaton with a *test automaton*. The testing process then consists in performing reachability analysis in the composed system restricted over all non internal actions. We say that the tested automaton fails the test if a special reject node of the test automaton is reachable in the parallel composition (restricted over all non internal actions) from their initial configurations, and passes otherwise.

### 3.1 What is precisely "Testing"?

The formal definition of testing involves the definition of what a test automaton is, how the parallel composition is performed and when the test has failed or succeeded. We now proceed to make these notions precise.

**Definition 3.1.** A *test automaton* is a tuple $T = \langle \mathsf{Act}_\tau, N, N_T, n_0, C, C_0, E\rangle$ where $\mathsf{Act}_\tau$, $N$, $n_0$, $C$, and $E$ are as in Definition 2.8, $N_T \subseteq N$ is the set of *reject nodes* and $C_0 \subseteq C$ is the set of clocks of the automaton whose value must be $0$ at the beginning of every run of the automaton. All the test automata we shall consider in this study will have either one reject state or none.

An *initial valuation* for $T$ is any valuation for the set of clocks $C$ that assigns the value $0$ to every clock in $C_0$. An *initial state* of $T$ is any state $\langle n_0, u_0\rangle$ of $T$ with $u_0$ an initial valuation.

**Convention.** In what follows, we shall assume that the clocks used in test automata come from a fixed, countably infinite collection of clocks $C_T$ disjoint from $C_A$.

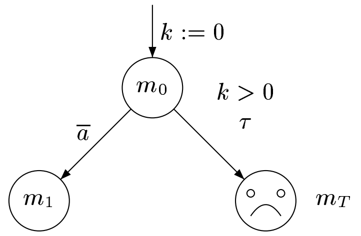
In the remainder of this paper, following the standard practice in the use of the model-checker UPPAAL, test automata will be used to reduce the verification of properties of states of TLTSs — other than plain reachability ones — to reachability analysis, as described in Section 1. Intuitively, a test automaton $T$ interacts with a tested system, represented by a TLTS, by communicating with it. In general, the initial state of the test automaton which is suitable for testing a given property will depend on the property itself. For instance, it is reasonable to expect that the tester for a property like "the value of the clock $x$ is equal to that of the clock $y$" will have $x$ and $y$ among its clocks, and that the current values of these clocks will be fed into the test automaton from its environment because they are necessary to determine the truth or falsity of the property in the current system state. (See Definition 3.4 for the precise definition of the property tested by a test automaton.) This is the reason why we allow a collection of possible initial states for test automata. Test automata may also use a collection of *private clocks — i.e.*, those included in the set $C_0$ —, and we stipulate that, as for timed automata, the initial values of such clocks should be 0.

Given a distinguished initial state for a test automaton $T$, the dynamics of the interaction between the tester and the tested system is described by the parallel composition of the TLTS that is being tested and of $\mathcal{T}_T$ (the TLTS describing the semantics of $T$), restricted over the set of actions Act. We now define failure and success of a test as follows.

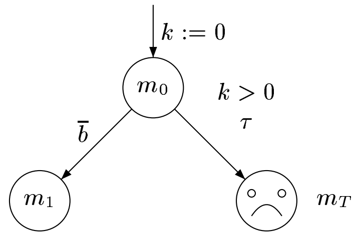
**Definition 3.2.** Let $\mathcal{T}$ be a TLTS, $T$ a test automaton and Act the set of actions.

- We say that a node $n$ of $T$ is reachable from a state $(s_1 \parallel s_2)\backslash\mathsf{Act}$ of $(\mathcal{T} \parallel \mathcal{T}_T)\backslash\mathsf{Act}$ iff there is a sequence of transitions leading from $(s_1 \parallel s_2)\backslash\mathsf{Act}$ to a state whose $\mathcal{T}_T$ component is of the form $\langle n, u\rangle$.
- We say that a state $s$ of $\mathcal{T}$ *fails the test* $T$ from the initial state $\langle n_0, u_0\rangle$ iff a reject node of $T$ is reachable in $(\mathcal{T} \parallel \mathcal{T}_T)\backslash\mathsf{Act}$ from the state $(s \parallel \langle n_0, u_0\rangle)\backslash\mathsf{Act}$. Otherwise, we say that $s$ *passes the test* $T$ from the initial state $\langle n_0, u_0\rangle$.

In the remainder of the paper, we shall often apply test automata to the TLTSs that give operational semantics to timed automata. In that case, we shall use the suggestive notation $(A \parallel T)\backslash\mathsf{Act}$ in lieu of $(\mathcal{T}_A \parallel \mathcal{T}_T)\backslash\mathsf{Act}$. The initial state of $T$ to be considered will always be clear from the context.




(a) Test automaton $T_a$ $(a \in \mathcal{U})$




(b) Test automaton $T_b$ $(b \notin \mathcal{U})$

**Figure 2.** The test automata $T_a$ and $T_b$

*Example 3.3.* Consider the timed automaton $A$ of Figure 1 and the test automaton $T_b$ $(b \notin \mathcal{U})$ of Figure 2(b), where we label the arrow coming into the initial node $m_0$ of $T_b$ with the assignment $k := 0$ to denote the fact that clock $k$ is contained in $C_0$. (This convention will be used throughout the paper.) The reject node $m_T$ of the test automaton is reachable from the initial state of $(A \parallel T_b)\backslash\mathsf{Act}$, as follows: as $b$ is not urgent, both automata can let a positive amount of time pass, thus enabling the $\tau$-transition from node $m_0$ in $T_b$ and making $m_T$ reachable. In this case we say that $A$ fails the test. If we test $A$ using the automaton $T_a$ ($a \in \mathcal{U}$) of Figure 2(a), then, in all cases, $A$ and $T_a$ must synchronize on $a$ and, since $a$ is urgent, no positive initial delay is possible. It follows that the reject node $m_T$ of $T_a$ is unreachable, and $A$ passes the test.

### 3.2 Testing Properties

We have just seen how we can perform tests on states of TLTSs. We now aim at using test automata to determine whether, given a set of properties $\mathbb{L}$ (interpreted over extended states of a TLTS, that is pairs $(s, u)$ where $s$ is a state of the TLTS and $u$ is a valuation of some given set of clocks), a given state of a TLTS satisfies a formula of $\mathbb{L}$. As already mentioned, this approach to model-checking for timed automata is not merely a theoretical curiosity, but it is the way in which model-checking of properties other than plain reachability ones is routinely carried out in a verification tool like UPPAAL.

**Definition 3.4 (Testing properties).** Let $\mathbb{L}$ be a set of properties, $\varphi$ be a formula in $\mathbb{L}$, and

$$T = \langle \mathsf{Act}_\tau, N, N_T, m_0, C, C_0, E \rangle$$

be a test automaton.

– For every extended state $(s, u)$ of a TLTS $\mathcal{T}$ ($u$ is a valuation over a set of clocks containing $C$), we say that $(s, u)$ passes the test $T$ iff no reject node of $T$ is reachable from the state $(s \parallel \langle m_0, [C_0 \to 0](u \upharpoonright C)\rangle)\backslash\mathsf{Act}$.
– We say that the test automaton $T$ *tests* for the formula $\varphi$ (and that $\varphi$ is *testable*) iff the following holds: for every TLTS $\mathcal{T}$ and every extended state $(s, u)$ of $\mathcal{T}$,

$$(s, u) \models \varphi \text{ iff } (s, u) \text{ passes the test } T \ . \tag{1}$$

  If (1) holds for arbitrary states of timed automata then we say that the test automaton $T$ *tests* for the formula $\varphi$ (and that $\varphi$ is *testable*) over states of timed automata.
  A property language is *testable* if every property of the language is testable.

As an immediate property of test automata, we get the following proposition:

**Proposition 3.5.** *Let $\mathbb{L}$ be a property language. If $\mathbb{L}$ is testable then for every TLTS $\mathcal{T}$, for every extended state $(s, u)$ of $\mathcal{T}$, for every property $\varphi \in \mathbb{L}$,*

$$(s, u) \models \varphi \ \textit{iff} \ \forall s'.\ s \xrightarrow{\tau}{}^{*} s' \implies (s', u) \models \varphi \ .$$

*Proof.* We just need to prove the only-if implication of the equivalence. Assume that $(s, u) \models \varphi$ but that there exists a state $s'$ such that $s \xrightarrow{\tau}{}^{*} s'$ and $(s', u) \not\models \varphi$. As $\mathbb{L}$ is testable, a reject node of $T_\varphi$ ($T_\varphi = \langle \mathsf{Act}_\tau, N, N_T, m_0, C, C_0, E\rangle$ is the test automaton for the formula $\varphi$) is reachable from the state $(s' \parallel \langle m_0, [C_0 \to 0](u \upharpoonright C)\rangle)\backslash\mathsf{Act}$. Thus, it is also the case that a reject node is reachable from the state $(s \parallel \langle m_0, [C_0 \to 0](u \upharpoonright C)\rangle)\backslash\mathsf{Act}$. As $\mathbb{L}$ is testable, this contradicts our assumption that $(s, u) \models \varphi$. □

## 4 Safety Modal Property Language

In the sequel, we will consider a dense-time property language with clocks, suitable for the specification of safety and bounded liveness properties of TLTSs. This property language is a fragment of the $L_\nu$ property language presented in [LLW95], taking into account the current distinction between urgent and non-urgent actions.

For the sake of clarity, we begin by presenting a first natural candidate for our property language, called SBLL, and show that it is testable (Sections 4.1 and 4.2). We then show that the formalism of test automata is strictly more expressive than SBLL (Section 4.3). The property language $L_{\forall S}$, for which an expressive completeness result will be given in Section 5, is then introduced in Section 4.4. Its expressiveness is studied in Sections 4.5 and 4.6. Finally, the testability of $L_{\forall S}$ is proven in Section 4.7.

### 4.1 The Property Language SBLL

**Definition 4.1 (The property language** SBLL**).** Let $K$ be a countably infinite set of clocks, disjoint from $C_A$ and including $C_T$. We use `fail` to denote an action symbol not contained in Act. The set SBLL of formulae over $K$ is generated by the following grammar:

$$\begin{array}{rcl} \varphi & ::= & \mathtt{ff} \;\mid\; \varphi_1 \wedge \varphi_2 \;\mid\; g \vee \varphi \;\mid\; \forall\!\!\forall \varphi \;\mid\; [a]\varphi \;\mid \\ & & \langle a \rangle \mathtt{tt}\;\; (a \in \mathcal{U}) \;\mid\; x\ \underline{\mathtt{in}}\ \varphi \;\mid\; X \;\mid\; \mathtt{max}(X, \varphi) \\ g & ::= & x \sim p \;\mid\; x - y \sim p \end{array}$$

where $a \in \mathsf{Act} \cup \{\mathtt{fail}\}$, $x, y \in K$, $p \in \mathbb{N}$, $\sim \in \{<, >, =\}$, $X$ is a formula variable and $\mathtt{max}(X, \varphi)$ stands for the maximal solution of the recursion equation $X = \varphi$.

A *closed recursive formula* of SBLL is a formula in which every occurrence of every formula variable $X$ appears within the scope of some $\mathtt{max}(X, \varphi)$ construct. Given a non closed formula, a variable that is not under the scope of some `max` operator is said *free*. We use $\mathrm{SBLL}^-$ to stand for the collection of closed recursive formulae in SBLL that do not contain occurrences of the basic propositions $\langle a \rangle \mathtt{tt}$. We use $\mathtt{clocks}(\varphi)$ to denote the collection of clocks occurring in the formula $\varphi$.

Following [LLW95], closed recursive formulae in SBLL are interpreted over extended states of TLTSs, *i.e.* over pairs of the form $(s, v)$, where $s$ is a state of a TLTS and $v$ is a valuation for the clocks in $K$. But, because of Proposition 3.5, the interpretation of SBLL defined as in [LLW95] is not suitable in our setting (the interpretation of the formulae must "be closed under silent actions"). We thus define the satisfaction relation for SBLL as the largest relation, denoted by $\models$, satisfying the implications in Table 3.

| | | |
|---|---|---|
| $(s, u) \models \mathtt{ff}$ | $\Rightarrow$ | $false$ |
| $(s, u) \models \varphi_1 \wedge \varphi_2$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi_1$ and $(s', u) \models \varphi_2$ |
| $(s, u) \models g \vee \varphi$ | $\Rightarrow$ | $g(u)$ or $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi$ |
| $(s, u) \models [a]\varphi$ | $\Rightarrow$ | $\forall s'.\ s \stackrel{a}{\Longrightarrow} s'$ implies $(s', u) \models \varphi$ |
| $(s, u) \models \langle a \rangle \mathtt{tt}\ (a \in \mathcal{U})$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $s' \xrightarrow{a} s''$ for some $s''$ |
| $(s, u) \models \forall\!\!\forall \varphi$ | $\Rightarrow$ | $\forall d \in \mathbb{R}_{\geq 0}\ \forall s'.\ s \stackrel{\epsilon(d)}{\Longrightarrow} s'$ implies $(s', u + d) \models \varphi$ |
| $(s, u) \models x\ \underline{\mathtt{in}}\ \varphi$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', [x \to 0]u) \models \varphi$ |
| $(s, u) \models \mathtt{max}(X, \varphi)$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi\{\mathtt{max}(X, \varphi)/X\}$ |

**Table 3.** Satisfaction implications for SBLL

### 4.2 Testing SBLL

Our goal is to use the property language that we just defined as a specification language and to reduce its model-checking to reachability testing. In order to achieve our goal, we shall define a "compilation" procedure to obtain a test automaton from the formula we want to test for. By means of this compilation procedure, we automate the process of generating test automata from logical specifications — a task which has so far required a high degree of ingenuity and is error-prone.

One of our first important results is that the property language SBLL is testable, in the sense that, for every closed recursive formula $\varphi \in$ SBLL, we can construct a test automaton $T_\varphi$ such that every extended state $(s, u)$ of a TLTS satisfies $\varphi$ iff it passes the test $T_\varphi$, in the sense of Definition 3.2. The following theorem states this result:

**Theorem 4.2.** *For every closed formula $\varphi$ in $\mathrm{SBLL}^-$, there exists a test automaton $T_\varphi$, over the set of clocks $\{k\} \cup \texttt{clocks}(\varphi)$, where $k$ does not occur in $\texttt{clocks}(\varphi)$, that tests for it.*

*For every closed formula $\varphi$ in $\mathrm{SBLL}$, there exists a test automaton $T_\varphi$, over the set of clocks $\{k\} \cup \texttt{clocks}(\varphi)$, where $k$ does not occur in $\texttt{clocks}(\varphi)$, that tests for it over states of timed automata.*

Before proving this theorem, we generalize our notion of testing to non-closed formulae, because it will be useful in the proof (which will be done by induction on the structure of the formulae). A *(non-closed) test automaton* is a tuple $T = \langle \mathsf{Act}_\tau, N, N_T, n_0, C, C_0, E, \mathcal{X}, \lambda\rangle$ where $\mathsf{Act}_\tau$, $N$, $N_T$ $n_0$, $C$, $C_0$ and $E$ are as in Definition 3.1, $\mathcal{X}$ is a set of variables and $\lambda : \mathcal{X} \longrightarrow 2^N$ is a labelling function, which associates to each variable a set of nodes in which "the variable is free". Of course, a simple test automaton can be viewed as a non-closed test-automaton such that the labelling function assigns to each variable the empty set.

Let $T$ be a test automaton (supposed non closed) and $T'$ be a test automaton (non-closed or not). We define the test automaton $T\{T'/X\}$ as the test automaton obtained from $T$ by replacing all the nodes labelled by $X$ by the automaton $T'$. The construction is depicted on Figure 3.

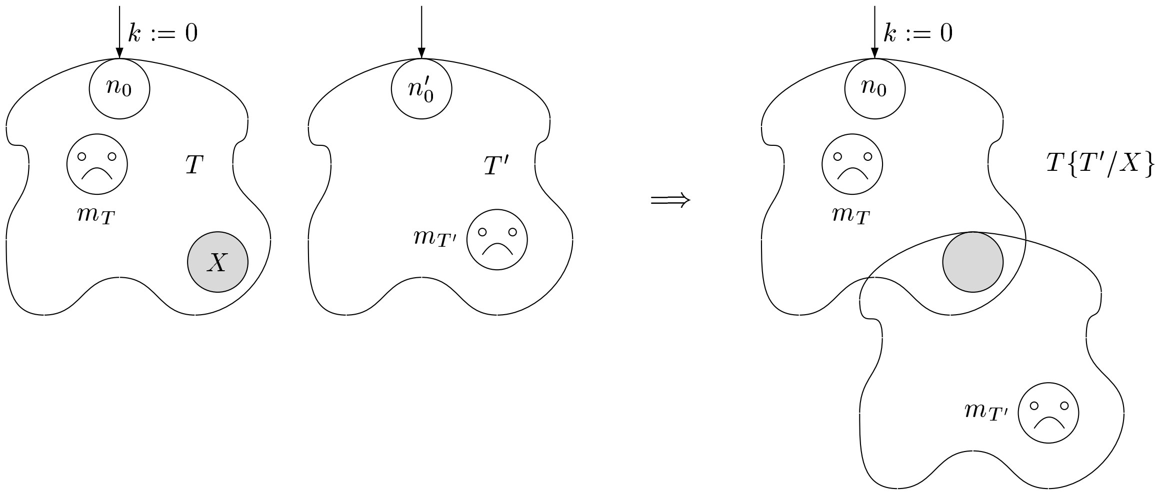
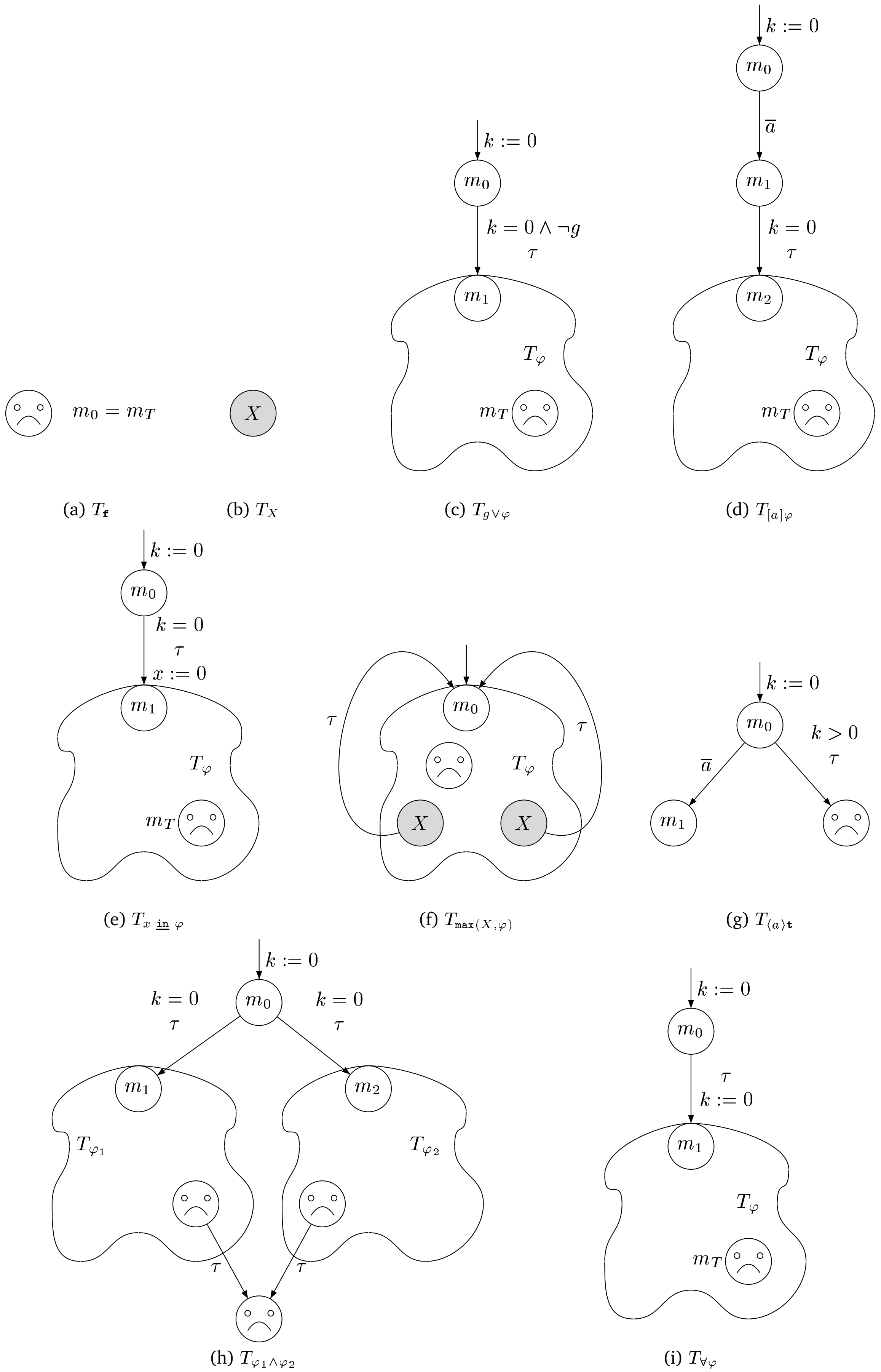



**Figure 3.** Construction of the test automaton $T\{T'/X\}$

We extend the definitions by defining testing for non-closed formulae: we say that a (non-closed) test automaton $T$ tests for a non-closed formula $\varphi$ whenever, for every testable properties $\varphi_1, \ldots, \varphi_k$, the closed test automaton $T\{T_{\varphi_1}/X_1, \ldots, T_{\varphi_k}/X_k\}$ tests for the closed formula $\varphi\{\varphi_1/X_1, \ldots, \varphi_k/X_k\}$ (we assume that $T_{\varphi_i}$ tests for $\varphi_i$ and that $\{X_1, \ldots, X_k\}$ is the set of free variables of $\varphi$).

*Proof.* Let us consider the automata $T_\psi$ depicted on Figure 4. We prove that they test for SBLL by a case analysis on the form of $\psi$.

- CASE $\psi = \mathtt{ff}$: the reject node of $T_{\mathtt{ff}}$ is reachable in any parallel composition of a TLTS and of $T_{\mathtt{ff}}$.
- CASE $\psi = g \vee \varphi$: let $\mathcal{T}$ be a TLTS. We first assume that a reject node of $T_{g\vee\varphi}$ is reachable from the initial state $(s_0 \parallel \langle m_0, [k_0 \to 0]w_0\rangle)\backslash\mathsf{Act}$ of $\mathcal{T}$ where $w_0$ is a given initial valuation. It




(a) $T_{\mathtt{f}}$ (b) $T_X$ (c) $T_{g \vee \varphi}$ (d) $T_{[a]\varphi}$

(e) $T_{x \ \underline{\mathtt{in}} \ \varphi}$ (f) $T_{\mathtt{max}(X,\varphi)}$ (g) $T_{\langle a \rangle \mathtt{t}}$

(h) $T_{\varphi_1 \wedge \varphi_2}$ (i) $T_{\forall \varphi}$

**Figure 4.** Test automata for SBLL subformulae

means that the following computation is allowed by the parallel composition:

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\xrightarrow{\tau}{}^{*} (s \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\xrightarrow{\tau} (s \parallel \langle m_1, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\leadsto^{*} (s' \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\end{aligned}$$

where, in particular, $(k = 0 \wedge \neg g)([k \to 0]w_0)$ is true and thus, $g(w_0)$ is false. By applying the induction hypothesis to $\varphi$, we also obtain that $(s, w_0) \not\models \varphi$. Thus, we conclude that $(s_0, w_0) \not\models g \vee \varphi$.
Conversely, assume that $(s_0, w_0) \not\models g \vee \varphi$: $g(w_0)$ is false and a reject node is reachable in the parallel composition $(\mathcal{T} \parallel T_\varphi)\backslash\mathsf{Act}$. Thus the following computation is allowed ($k$ is a new clock not appearing in $T_\varphi$):

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\xrightarrow{\tau} (s_0 \parallel \langle m_1, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\leadsto^{*} (s' \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\ .\end{aligned}$$

Thus, the reject node of $T_{g\vee\varphi}$ is reachable from $s_0$ in the parallel composition $(\mathcal{T} \parallel T_{g\vee\varphi})\backslash\mathsf{Act}$.

- CASE $\psi = [a]\varphi$ : let $\mathcal{T}$ be a TLTS. We first assume that a reject node of $T_{[a]\varphi}$ is reachable from $(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}$. Thus the following computation is allowed:

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\xrightarrow{\tau}{}^{*} (s \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\xrightarrow{\tau=a/\overline{a}} (s' \parallel \langle m_1, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\xrightarrow{\tau}{}^{*} (s'' \parallel \langle m_1, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\xrightarrow{\tau} (s'' \parallel \langle m_2, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\leadsto^{*} (s''' \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\ .\end{aligned}$$

This means precisely that $s_0 \stackrel{a}{\Longrightarrow} s''$ and that $(s'', [k \to 0]w_0) \not\models \varphi$, thus $(s_0, [k \to 0]w_0) \not\models [a]\varphi$.
Conversely, assume that $(s_0, [k \to 0]w_0]) \not\models [a]\varphi$: there exists some state $s_1$ such that $s_0 \stackrel{a}{\Longrightarrow} s_1$ and $(s_1, [k \to 0]w_0) \not\models \varphi$. Thus, in the parallel composition $(\mathcal{T} \parallel T_{[a]\varphi})\backslash\mathsf{Act}$,

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} \xrightarrow{\tau}{}^{*} \xrightarrow{\tau=a/\overline{a}} &(s_1 \parallel \langle m_1, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ \xrightarrow{\tau}\quad &(s_1 \parallel \langle m_2, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ \leadsto^{*}\quad &(s_2 \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\end{aligned}$$

which means that a reject node is reachable from $(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}$.

- CASE $\psi = x\ \underline{\mathtt{in}}\ \varphi$: let $\mathcal{T}$ be a TLTS. We first assume that a reject node of $T_{x\ \underline{\mathtt{in}}\ \varphi}$ is reachable from $(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}$. The following computation is allowed:

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\xrightarrow{\tau}{}^{*} (s \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\xrightarrow{\tau} (s \parallel \langle m_1, [k, x \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\leadsto^{*} (s' \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\ .\end{aligned}$$

This means precisely that $(s, [x, k \to 0]w_0) \not\models \varphi$, thus $(s_0, [k \to 0]w_0) \not\models x\ \underline{\mathtt{in}}\ \varphi$.
Conversely, assume that $(s_0, [k \to 0]w_0) \not\models x\ \underline{\mathtt{in}}\ \varphi$: $(s_0, [k, x \to 0]w_0) \not\models \varphi$. Thus, in the parallel composition $(\mathcal{T} \parallel T_{x\ \underline{\mathtt{in}}\ \varphi})\backslash\mathsf{Act}$,

$$\begin{aligned}(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\xrightarrow{\tau} (s_0 \parallel \langle m_1, [k, x \to 0]w_0\rangle)\backslash\mathsf{Act}\\ &\leadsto^{*} (s \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act}\end{aligned}$$

which means that a reject node is reachable from $(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act}$.

- CASE $\psi = X$: It is immediate that the non-closed test automaton depicted on Figure 4(b) tests for the formula $X$ (in the sense defined just before the proof).

– CASE $\psi = \max(X, \varphi)$: It is difficult to prove the property directly on the recursive formula, thus we define inductively the following formulae:

$$\begin{array}{ll} \psi^{(0)} & = \varphi\{\mathtt{tt}/X\} \\ \psi^{(l+1)} & = \varphi\{\psi^{(l)}/X\} \end{array}$$

Let $\mathcal{T}$ be a TLTS. Let $(s_0, u)$ be an extended state of $\mathcal{T}$ ($s_0$ is the initial node of $\mathcal{T}$). As an easy property of maximal fixed points [Tar55], we have that

$$(s_0, u) \models \max(X, \varphi) \iff \forall l \geq 0.\ (s_0, u) \models \psi^{(l)} \ .$$

The test automata for $\psi^{(l)}$ are obtained by a sequential juxtaposition of $l$ copies of $T_\varphi$, more precisely $T_{\psi^{(0)}}$ is $T_\varphi\{T_{\mathtt{tt}}/X\}$ and $T_{\psi^{(l+1)}}$ is $T_\varphi\{T_{\psi^{(l)}}/X\}$ where $T_{\mathtt{tt}}$ is the test automaton with a single (initial) node and with no rejecting node.
Thus, we just have to prove that a reject node of $T_{\max(X,\varphi)}$ is reachable in the parallel composition $(\mathcal{T} \parallel T_{\max(X,\varphi)})\backslash\mathsf{Act}$ if and only if, for some integer $l$, a reject node of $T_{\psi^{(l)}}$ is reachable in the parallel composition $(\mathcal{T} \parallel T_{\psi^{(l)}})\backslash\mathsf{Act}$.
To this aim, we define for each integer $l$ a projection $\pi_l$ from $T_{\psi^{(l)}}$ onto $T_{\max(X,\varphi)}$ such that, if $m$ is a state of $T_{\psi^{(l)}}$, $\pi_l(m)$ is the "same" state in $T_{\max(X,\varphi)}$ (recall that $T_{\psi^{(l)}}$ is a juxtaposition of copies of $T_\varphi$ and that $T_{\mathtt{max}(X,\varphi)}$ has the same set of states than $T_\varphi$) and $\pi_l(m \xrightarrow{\alpha} m')$ is $\pi_l(m) \xrightarrow{\alpha} \pi_l(m')$. We have the two following properties:

$$\begin{array}{rcl} \langle m, u\rangle \xrightarrow{\alpha} \langle m', u'\rangle \text{ in } T_{\psi^{(l)}} & \Longrightarrow & \langle \pi_l(m), u\rangle \xrightarrow{\alpha} \langle \pi_l(m'), u'\rangle \text{ in } T_{\mathtt{max}(X,\varphi)} \\ & \text{and} & \\ m \text{ reject node of } T_{\psi^{(l)}} & \iff & \pi_l(m) \text{ reject node of } T_{\mathtt{max}(X,\varphi)} \ . \end{array}$$

If

$$\langle m_{i_0}, u_0\rangle \xrightarrow{\alpha_1} \langle m_{i_1}, u_1\rangle \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_p} \langle m_{i_p}, u_p\rangle$$

is a computation in $T_{\psi^{(l)}}$, then

$$\langle \pi_l(m_{i_0}), u_0\rangle \xrightarrow{\alpha_1} \langle \pi_l(m_{i_1}), u_1\rangle \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_p} \langle \pi_l(m_{i_p}), u_p\rangle$$

is a computation in $T_{\max(X,\varphi)}$. Thus, if a reject node is reachable in the parallel composition $(\mathcal{T} \parallel T_{\psi^{(l)}})\backslash\mathsf{Act}$, then a reject node is reachable in the parallel composition $(\mathcal{T} \parallel T_{\max(X,\varphi)})\backslash\mathsf{Act}$.
Conversely, consider a computation in $T_{\mathtt{max}(X,\varphi)}$

$$\langle m_{i_0}, u_0\rangle \xrightarrow{\alpha_1} \langle m_{i_1}, u_1\rangle \ldots \xrightarrow{\alpha_p} \langle m_{i_p}, u_p\rangle \tag{2}$$

and define $J$ the following set of indexes: $J = \{j \in \{1, \ldots, p\} \mid \text{the transition } \xrightarrow{\alpha_j} \text{ is new}\}$ where a transition is said *new* whenever it is in the test automaton $T_{\mathtt{max}(X,\varphi)}$ whereas it is not in $T_\varphi$ (*i.e.* it is a transition $m \xrightarrow{\tau} m_0$ where $m$ is labelled by $X$ whereas $m_0$ is initial in $T_\varphi$). Define $l$ by $l = \#J + 1$ where $\#J$ denotes the cardinality of $J$. We will prove that there exists a run in $T_{\psi^{(l)}}$

$$\langle m'_{i_0}, u_0\rangle \xrightarrow{\alpha_1} \langle m'_{i_1}, u_1\rangle \ldots \xrightarrow{\alpha_p} \langle m'_{i_p}, u_p\rangle$$

whose image by $\pi_l$ is the run (2).
Assume $J = \{j_1, \ldots j_{l-1}\}$ with $j_i < j_{i+1}$ for each $1 \leq i < l-1$. For each $1 \leq i < l-1$, for each $j_i \leq h < j_{i+1}$, the state $m'_{i_h}$ is defined as the single state of the $i^{th}$ copy of $T_\varphi$ in $T_{\mathtt{max}(X,\varphi)}$ such that $\pi_l(m'_{i_h}) = m_{i_h}$. The run

$$\langle m'_{i_0}, u_0\rangle \xrightarrow{\alpha_1} \langle m'_{i_1}, u_1\rangle \ldots \xrightarrow{\alpha_p} \langle m'_{i_p}, u_p\rangle$$

is a run of $T_{\psi^{(l)}}$ whose image by $\pi_l$ is (2).
We thus proved: if a reject node is reachable in the parallel composition $(\mathcal{T} \parallel T_{\max(X,\varphi)})\backslash\mathsf{Act}$, then a reject node is reachable in the parallel composition $(\mathcal{T} \parallel T_{\psi^{(l)}})\backslash\mathsf{Act}$ for some integer $l$.

– CASE $\psi = \langle a \rangle \mathtt{tt}$: let $A$ be a timed automaton. We first assume that a reject node of $T_{\langle a \rangle \mathtt{tt}}$ is reachable from the initial state $\langle n_0, u_0 \rangle$ of $A$ if we put in parallel $A$ and $T_{\langle a \rangle \mathtt{tt}}$ with $w_0$ as initial valuation for $T_{\langle a \rangle \mathtt{tt}}$. It means that the following computation is allowed by the parallel composition $(A \parallel T_{\langle a \rangle \mathtt{tt}}) \backslash \mathsf{Act}$: there exists some reals $d, d' > 0$ such that

$$\begin{aligned} (\langle n_0, u_0 \rangle \backslash \mathsf{Act} \parallel \langle m_0, [k \to 0] w_0 \rangle) &\xrightarrow{\tau}^* (\langle n_1, u_1 \rangle \parallel \langle m_0, [k \to 0] w_0 \rangle) \backslash \mathsf{Act} \\ &\xrightarrow{\epsilon(d')} (\langle n_2, u_2 \rangle \parallel \langle m_0, ([k \to 0] w_0) + d' \rangle) \backslash \mathsf{Act} \\ &\stackrel{\epsilon(d-d')}{\Longrightarrow} (\langle n_3, u_3 \rangle \parallel \langle m_0, ([k \to 0] w_0) + d \rangle) \backslash \mathsf{Act} \\ &\xrightarrow{\tau} (\langle n_3, u_3 \rangle \parallel \langle m_T, ([k \to 0] w_0) + d \rangle) \backslash \mathsf{Act} \ . \end{aligned}$$

Moreover, as $a$ is urgent (and thus $\overline{a}$ is urgent), it means that $\langle n_1, u_1 \rangle \stackrel{a}{\not\longrightarrow}$ and thus that $(\langle n_0, u_0 \rangle, [k \to 0] w_0) \not\models \langle a \rangle \mathtt{tt}$.
Conversely, assume that $(\langle n_0, u_0 \rangle, w_0) \not\models \langle a \rangle \mathtt{tt}$. It means that there exists some extended state of $A$, $\langle n_1, u_1 \rangle$ such that $\langle n_0, u_0 \rangle \xrightarrow{\tau}^* \langle n_1, u_1 \rangle$ and $\langle n_1, u_1 \rangle \stackrel{a}{\not\longrightarrow}$. Thus, the following computation is allowed in the parallel composition:

$$\begin{aligned} (\langle n_0, u_0 \rangle \parallel \langle m_0, [k \to 0] w_0 \rangle) \backslash \mathsf{Act} &\xrightarrow{\tau}^* (\langle n_1, u_1 \rangle \parallel \langle m_0, [k \to 0] w_0 \rangle) \backslash \mathsf{Act} \\ &\xrightarrow{\epsilon(1)} (\langle n_1, u_1 + 1 \rangle \parallel \langle m_0, ([k \to 0] w_0) + 1 \rangle) \backslash \mathsf{Act} \\ &\xrightarrow{\tau} (\langle n_1, u_1 + 1 \rangle \parallel \langle m_T, ([k \to 0] w_0) + 1 \rangle) \backslash \mathsf{Act} \ . \end{aligned}$$

Thus, a reject node is reachable in $(A \parallel T_{\langle a \rangle \mathtt{tt}}) \backslash \mathsf{Act}$.

– CASE $\psi = \varphi_1 \wedge \varphi_2$: this case is very easy and the proof is omitted.

– CASE $\psi = \forall\!\!\!\forall \varphi$: let $\mathcal{T}$ be a TLTS. Assume that a reject node is reachable from

$$(s_0 \parallel \langle m_0, [k \to 0] w_0 \rangle) \backslash \mathsf{Act}$$

in the parallel composition $(\mathcal{T} \parallel T_{\forall\!\!\!\forall \varphi}) \backslash \mathsf{Act}$. The following computation is allowed ($k$ is a clock not appearing in $T_\varphi$): there exists some real $d \geq 0$ such that

$$\begin{aligned} (s_0 \parallel \langle m_0, [k \to 0] w_0 \rangle) \backslash \mathsf{Act} &\stackrel{\epsilon(d)}{\Longrightarrow} (s_1 \parallel \langle m_0, ([k \to 0] w_0) + d \rangle) \backslash \mathsf{Act} \\ &\xrightarrow{\tau} (s_1 \parallel \langle m_1, ([k \to 0](w_0 + d)) \rangle) \backslash \mathsf{Act} \\ &\rightsquigarrow^* (s_2 \parallel \langle m_T, w_T \rangle) \backslash \mathsf{Act} \ . \end{aligned} \tag{3}$$

In particular, $(s_1, w_0 + d) \not\models \varphi$ ($k$ is not useful for $\varphi$). Thus, $(s_0, [k \to 0] w_0) \not\models \forall\!\!\!\forall \varphi$.
Conversely, assume that $(s_0, w_0) \not\models \forall\!\!\!\forall \varphi$. Then a computation like (3) is allowed by the parallel composition $(\mathcal{T} \parallel T_{\forall\!\!\!\forall \varphi}) \backslash \mathsf{Act}$: a reject node is thus reachable.

This concludes the proof. □

It is now natural to wonder whether every property $\varphi$ that is testable in this fashion can be expressed in the property language SBLL. This amounts to asking whether every test automaton $T$ is expressible in the language SBLL, in the sense that there exists a formula $\psi_T$ of SBLL such that every timed automaton $A$ passes the test $T$ iff $A$ satisfies $\psi_T$. Indeed, an answer to this question would allow us to completely characterize the class of properties of timed automata for which the model-checking problem can be effectively reduced to deciding reachability of states in test automata, as it is commonly done in the verification tool UPPAAL. (See, *e.g.*, the references [BGK+96, JLS96, KP95] for examples of applications of this approach to verification in UPPAAL.)

### 4.3 SBLL is not Sufficient!

The starting point of our current investigation is the realization that test automata have a greater expressive power than the specification language SBLL. As an example, consider the test automaton $T$ depicted in Figure 5, where $a$ is an urgent action.

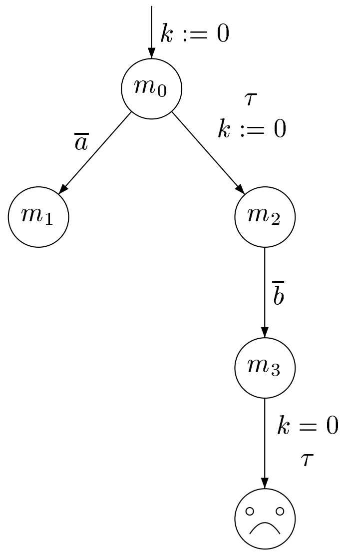



**Figure 5.** A test automaton, $T$, that cannot be expressed in SBLL ($a \in \mathcal{U}$)

We shall soon see (see the proof of Theorem 4.13) that there are two timed automata which satisfy the same properties expressible in the language SBLL, but that behave in different ways when exposed to the test $T$. In particular, one of them passes the test $T$, in the sense of Definition 3.2, whereas the other does not. This shows that the property tested by the automaton in Figure 5 is *not* expressible in the language SBLL. Intuitively, the property that is tested by the automaton in Figure 5 requires that, by delaying without enabling an $a$ action in the process, a state can only evolve to one in which it can*not* perform the action $b$. This is the import of the following observation, whose proof will give the reader a taste of some of the arguments to follow.

**Proposition 4.3.** *Let $s$ be a state in a TLTS $\mathcal{T}$. Then $s$ passes the test in Figure 5 iff for every state $s'$ of $\mathcal{T}$ and delay $d \in \mathbb{R}_{\geq 0}$, $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ implies $s' \stackrel{b}{\not\Longrightarrow}$.*

*Proof.* See Appendix A for the proof of this proposition. □

The kind of test automaton depicted in Figure 5 suggests an enrichment of the property language SBLL in which the delay construct $\forall\!\!\!\forall$ is parameterized by a set of urgent actions, whose elements should not become enabled as a state delays. It is perhaps surprising that, as we shall show in the sequel (see Theorem 5.8), this simple extension of SBLL yields a property language that is expressive complete with respect to the collection of reachability properties expressible by means of test automata, in the sense of Definition 3.2.

### 4.4 An Enrichment of SBLL, the Property Language $L_{\forall S}$

The property language we study here is an extension of SBLL (see Definition 4.1), and is closely related to the modal logic $L_\nu$ presented in [LLW95], and further investigated in [LL95]. The complexity of the model-checking of such a property language has been studied in [AL99] and $L_\nu$ is used as a specification language in the tool CMC [LL98].

**Definition 4.4.** Let $K$ be a countably infinite set of clocks, disjoint from $C_A$ and including $C_T$. We use `fail` to denote an action symbol not contained in Act. The set $L_{\forall S}$ of formulae over $K$ is generated by the following grammar:

$$
\begin{array}{rcl}
\varphi & ::= & \mathtt{ff} \ \mid\ \varphi_1 \wedge \varphi_2 \ \mid\ g \vee \varphi \ \mid\ \forall\!\!\!\forall_S \varphi \ \mid\ [a]\varphi \ \mid \\
 & & \langle a \rangle \mathtt{tt}\ \ (a \in \mathcal{U}) \ \mid\ x\ \underline{\mathtt{in}}\ \varphi \ \mid\ X \ \mid\ \mathtt{max}(X, \varphi) \\
g & ::= & x \sim p \ \mid\ x - y \sim p
\end{array}
$$

where $S \subseteq \mathcal{U}$, $a \in \mathsf{Act} \cup \{\mathtt{fail}\}$, $x, y \in K$, $p \in \mathbb{N}$, $\sim \in \{<, >, =\}$, $X$ is a formula variable and $\mathtt{max}(X, \varphi)$ stands for the maximal solution of the recursion equation $X = \varphi$. We use $L^{-}_{\forall S}$ to stand for the collection of formulae in $L_{\forall S}$ that do not contain occurrences of the basic propositions $\langle a \rangle \mathtt{tt}$.

As for SBLL, we define a *closed recursive formula* of $L_{\forall S}$ as a formula in which every occurrence of every formula variable $X$ appears within the scope of some $\mathtt{max}(X, \varphi)$ construct. In the remainder of this paper, every formula will be closed, unless specified otherwise. As for SBLL, we use $\mathtt{clocks}(\varphi)$ to denote the collection of clocks occurring in the formula $\varphi$. A clock $x$ is free in the formula $\phi$ if there is some occurrence of $x$ in $\phi$ which is not within the scope of some $x \ \underline{\mathtt{in}} \ \varphi$ construct. A formula is *clock closed* if it has no free clock variable.

Given a TLTS $\mathcal{T} = \langle S, \mathcal{L}, s^0, \longrightarrow \rangle$, we interpret, as usual, the closed formulae in $L_{\forall S}$ over extended states. We recall that an *extended state* is a pair $(s, u)$ where $s$ is a state of $\mathcal{T}$ and $u$ is a time assignment for the formula clocks in $K$. The satisfaction relation $\models$ is the largest relation satisfying the implications in Table 4. A relation satisfying these implications is called a *satisfiability relation*.

| | | |
|---|---|---|
| $(s,u) \models \mathtt{ff}$ | $\Rightarrow$ | $false$ |
| $(s,u) \models \varphi_1 \wedge \varphi_2$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi_1$ and $(s', u) \models \varphi_2$ |
| $(s,u) \models g \vee \varphi$ | $\Rightarrow$ | $g(u)$ or $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi$ |
| $(s,u) \models [a]\varphi$ | $\Rightarrow$ | $\forall s'.\ s \stackrel{a}{\Longrightarrow} s'$ implies $(s', u) \models \varphi$ |
| $(s,u) \models \langle a \rangle \mathtt{tt}$ $(a \in \mathcal{U})$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $s' \xrightarrow{a} s''$ for some $s''$ |
| $(s,u) \models \forall_S \varphi$ | $\Rightarrow$ | $\forall d \in \mathbb{R}_{\geq 0}\ \forall s'.\ s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$ implies $(s', u + d) \models \varphi$ |
| $(s,u) \models x \ \underline{\mathtt{in}} \ \varphi$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', [x \to 0]u) \models \varphi$ |
| $(s,u) \models \mathtt{max}(X, \varphi)$ | $\Rightarrow$ | $\forall s'.\ s \xrightarrow{\tau}{}^{*} s'$ implies $(s', u) \models \varphi\{\mathtt{max}(X, \varphi)/X\}$ |

**Table 4.** Satisfaction implications for $L_{\forall S}$

It follows from standard fixed-point theory [Tar55] that $\models$ is the union of all satisfiability relations and that the above implications are in fact biimplications for $\models$.

We say that $\mathcal{T}$ satisfies $\varphi$, written $\mathcal{T} \models \varphi$, when $(s^0, K \to 0) \models \varphi$. In the sequel, for a timed automaton $A$, we shall write $A \models \varphi$ in lieu of $\mathcal{T}_A \models \varphi$.

**Remark.** Since $\mathtt{fail}$ is not contained in Act, every extended state of a TLTS trivially satisfies formulae of the form $[\mathtt{fail}]\phi$. The role played by these formulae in the developments of this paper will become clear in Section 5.

The following lemma states a basic sanity property of the satisfaction relation.

**Lemma 4.5.** *If $u$ and $v$ are two valuations agreeing on the free clock variables in a formula $\varphi$, then, for every state $s$ of a TLTS,*

$$(s,u) \models \varphi \ \textit{ implies } \ (s,v) \models \varphi \ .$$

*Proof.* The relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \{((s,v), \varphi) \mid (s,u) \models \varphi \text{ and } u, v \text{ agree on the free clock variables of } \varphi\}$$

is a satisfiability relation. The straightforward verification is left to the reader. □

If the formula $\phi$ is clock closed, then the truth or falsity of the statement $(s, u) \models \phi$ is independent of the valuation $u$ (Lemma 4.5). In that case, we shall often simply write $s \models \phi$ in lieu of $(s, u) \models \phi$. It is also clear that the truth or falsity of the statement $(s, u) \models \phi$ depends at most on the restriction of $u$ to the set of clocks occurring in $\varphi$. Hence, in proofs in which we combine assignments, we shall often write $(s, u \restriction \mathtt{clocks}(\varphi)) \models \varphi$ in lieu of $(s, u) \models \varphi$.

The satisfaction relation is closed with respect to the relation $\xrightarrow{\tau}^*$, in the sense of the following proposition.

**Proposition 4.6.** *Let $\mathcal{T} = \langle \mathcal{S}, \mathcal{L}, s^0, \longrightarrow \rangle$ be a TLTS. Then, for every $s \in \mathcal{S}$, $\varphi \in L_{\forall S}$ and for every valuation $u$ for the clocks in $K$, $(s, u) \models \varphi$ iff for every $s'$ such that $s \xrightarrow{\tau}^* s'$, $(s', u) \models \varphi$.*

*Proof.* The only interesting thing to check is that if $(s, u) \models \varphi$ and $s \xrightarrow{\tau}^* s'$, then $(s', u) \models \varphi$. To this end, it is sufficient to prove that the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \{((s, u), \varphi) \mid \exists t.\ (t, u) \models \varphi \text{ and } t \xrightarrow{\tau}^* s\} \cup \models$$

is a satisfiability relation. The straightforward verification is left to the reader. □

This proposition meets the requirements of Proposition 3.5.

The reader familiar with the literature on variations on Hennessy-Milner logic [Mil89] and on its real-time extensions [Yi91] may have noticed that our definition of the satisfaction relation is rather different from the standard one presented in the literature. For instance, one might expect the clause of the definition of the satisfaction relation for the formula $\langle a \rangle \mathtt{tt}$ to read

$$(s, u) \models \langle a \rangle \mathtt{tt} \quad \text{implies} \quad s \stackrel{a}{\Longrightarrow} s' \text{ for some } s' \ . \tag{4}$$

Recall, however, that our main aim in this paper is to develop a specification language for timed automata for which the model-checking problem can be effectively reduced to deciding reachability. With this aim in mind, a reasonable proposal for a test automaton for the formula $\langle a \rangle \mathtt{tt}$, interpreted as in (4), is the automaton depicted in Figure 2(a). However, it is not hard to see that such an automaton could be brought into its reject node $m_T$ by one of its possible interactions with the timed automaton associated with the TCCS agent $a + \tau$. This is due to the fact that, because of the definition of parallel composition we have chosen, a test automaton cannot prevent the tested state from performing its internal transition leading to a state where an $a$-action is no longer possible.

### 4.5 Basic Properties of $L_{\forall S}$

**Definition 4.7.** Two formulae $\phi$ and $\psi$ in $L_{\forall S}$ are said to be equivalent over states of TLTSs (respectively, states of timed automata) iff, for every state $s$ of a TLTS (resp. of a timed automaton) and valuation $w$ for the formula clocks, $(s, w) \models \phi$ iff $(s, w) \models \psi$.

Let $\mathbb{L}$ and $\mathbb{L}'$ be two subsets of $L_{\forall S}$. We say that $\mathbb{L}$ is *at least as expressive as* $\mathbb{L}'$ over states of TLTSs (respectively, states of timed automata) iff for every formula $\phi$ in $\mathbb{L}'$, there exists a formula $\psi$ in $\mathbb{L}$ which is equivalent to it over states of TLTSs (respectively, states of timed automata).

If $\mathbb{L}$ is at least as expressive as $\mathbb{L}'$ over states of TLTSs (respectively, states of timed automata), and there is a formula $\phi$ in $\mathbb{L}$ which is not equivalent to any formula in $\mathbb{L}'$ over states of TLTSs (respectively, states of timed automata), then we say that $\mathbb{L}$ is *strictly more expressive than* $\mathbb{L}'$ over states of TLTSs (respectively, states of timed automata).

The languages $\mathbb{L}$ and $\mathbb{L}'$ are said to be *equally expressive* over states of TLTSs (respectively, states of timed automata) iff $\mathbb{L}$ is at least as expressive as $\mathbb{L}$ over states of TLTSs (respectively, states of timed automata) and vice versa.

Since, for every timed automaton $A$, the structure $\mathcal{T}_A$ is a TLTS (Proposition 2.12), it follows that if two formulae are equivalent over states of TLTSs, then they are also equivalent over states of timed automata. We shall see in what follows that the converse fails.

The following lemma collects a few basic results on equivalence between formulae, some of which will be useful in some of the proofs in Sections 5 and 6.

**Lemma 4.8.**

1. *For every formula $\varphi$ and clocks $x, y$, the formula $x \ \underline{\mathtt{in}}\ (y \ \underline{\mathtt{in}}\ \varphi)$ is equivalent over states of TLTSs to $y \ \underline{\mathtt{in}}\ (x \ \underline{\mathtt{in}}\ \varphi)$.*
2. *For every formula $\varphi$ and set of urgent actions $S$, the formula $\forall\!\!\!\forall_S \varphi$ is equivalent over states of TLTSs to $\forall\!\!\!\forall_S \forall\!\!\!\forall_S \varphi$.*
3. *For formulae $\varphi, \phi$ and action $a$, the formula $[a](\varphi \wedge \phi)$ is equivalent over states of TLTSs to $([a]\varphi) \wedge ([a]\phi)$.*

**Notation.** For a set of formula clocks $\{y_1, \ldots, y_n\}$ and a formula $\varphi$, we write $\{y_1, \ldots, y_n\} \ \underline{\mathtt{in}}\ \varphi$ as a short-hand for $y_1 \ \underline{\mathtt{in}}\ (y_2 \ \underline{\mathtt{in}}\ \cdots (y_n \ \underline{\mathtt{in}}\ \varphi) \cdots)$. (This notation is justified by Lemma 4.8(1) above.) If $n = 0$, then, by convention, $\emptyset \ \underline{\mathtt{in}}\ \varphi$ stands for $\varphi$. We also write $g$ in lieu of $g \vee \mathtt{ff}$, and $\mathtt{tt}$ for $x \geq 0$. When $\phi \in \mathcal{B}(K)$ and $\psi \in L_{\forall S}$, we use $\phi \Rightarrow \psi$ for $\neg\phi \vee \psi$, where $\neg\phi$ stands for the guard which is the negation of $\phi$.

*Example 4.9.* Consider the (template for a) timed automaton in Figure 6, where $a$ is an urgent action.

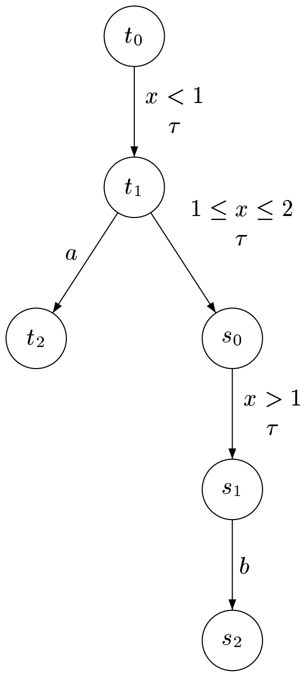



**Figure 6.** Template for a timed automaton ($a \in \mathcal{U}$)

We claim that:

1. $\langle t_0, [x \to 0] \rangle \models \forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$, but
2. $\langle s_0, [x \to 0] \rangle \not\models \forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$.

We begin by arguing that $\langle t_0, [x \to 0] \rangle$ satisfies the formula $\forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$. To this end, assume that $\langle t_0, [x \to 0] \rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} t$ for some state $t$. We proceed to show that $t \stackrel{b}{\not\Longrightarrow}$ by examining the possible forms $t$ may take. We distinguish two cases depending on whether $d$ is strictly smaller than 1 or not.

- Assume that $d < 1$. Then either $t = \langle t_0, [x \to d] \rangle$ or $\langle t_1, [x \to d] \rangle$. In both cases, since $d < 1$, the $\tau$-labelled edge from $t_1$ is not enabled, and therefore we may infer that $t \stackrel{b}{\not\Longrightarrow}$.

– Assume that $d \geq 1$. We claim that $t$ can only be $\langle t_0, [x \to d]\rangle$ — yielding the desired result because, as $d \geq 1$, no action can be performed from such a state. To see that this must indeed hold, note first of all that $\langle t_0, [x \to 0]\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} \langle t_0, [x \to d]\rangle$ because the only outgoing edge from $t_0$ is labelled by $\tau$. Moreover, since $d \geq 1$, any transition of the form $\langle t_0, [x \to 0]\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} t$ which leaves the node $t_0$ at some point must have the following prefix for some $0 \leq d' < 1$:

$$\langle t_0, [x \to 0]\rangle \stackrel{\epsilon(d')}{\longrightarrow}_{\{a\}} \langle t_0, [x \to d']\rangle \stackrel{\tau}{\longrightarrow} \langle t_1, [x \to d']\rangle \ .$$

Since $\langle t_1, [x \to d']\rangle \stackrel{a}{\longrightarrow}$ and $d' < 1$, the positive delay that is needed to enable the $\tau$-labelled transition from node $t_1$ cannot be performed without meeting an $a$-action. We may therefore conclude that, as claimed, no transition of the form $\langle t_0, [x \to 0]\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} t$ with $d \geq 1$ can leave node $t_0$.

On the other hand, $\langle s_0, [x \to 0]\rangle \not\models \mathbb{\forall}_{\{a\}}[b]\mathtt{ff}$. In fact, for instance,

$$\langle s_0, [x \to 0]\rangle \stackrel{\epsilon(1.1)}{\Longrightarrow}_{\{a\}} \langle s_1, [x \to 1.1]\rangle \stackrel{b}{\longrightarrow} \ .$$

The above example is of particular interest because, as we shall soon see, every property expressible in the language SBLL which is satisfied by the state $\langle t_0, [x \to 0]\rangle$ is also satisfied by $\langle s_0, [x \to 0]\rangle$.

### 4.6 Expressiveness Results for $L_{\forall S}$ and $L^-_{\forall S}$

In this section, we shall study the relative expressive power of the property languages introduced previously over the two classes of models that we consider in this paper, *viz.* states of timed automata and states of arbitrary TLTSs. We begin by arguing that the languages $L_{\forall S}$ and $L^-_{\forall S}$ can express exactly the same properties of states of timed automata, but that $L_{\forall S}$ is strictly more expressive than $L^-_{\forall S}$ over states of arbitrary TLTSs. As we shall see in the sequel, the extra expressive power of $L_{\forall S}$ over such models is obtained at the price of adding a property, *viz.* $\langle a\rangle\mathtt{tt}$, to $L^-_{\forall S}$ which can*not* be tested (over states of TLTSs) in the sense of this paper.

**Theorem 4.10.** *The property languages $L_{\forall S}$ and $L^-_{\forall S}$ are equally expressive over states of timed automata, but the former is strictly more expressive than the latter over states of arbitrary TLTSs.*

Since $L^-_{\forall S}$ is a sub-language of $L_{\forall S}$, it is clear that every property of states of arbitrary TLTSs (and, *a fortiori*, of states of timed automata) that can be expressed in the former can also be expressed in the latter. We shall now show that the converse holds over states of timed automata. Since the only extra constructs of $L_{\forall S}$ are those of the form $\langle a\rangle\mathtt{tt}$ ($a \in \mathcal{U}$), it is sufficient to show that these constructs can already be encoded into $L^-_{\forall S}$ over the class of models given by states of timed automata. This is the import of the following result.

**Lemma 4.11.**

1. *Let $s$ be a state of a TLTS, and $w$ be a valuation for the formula clocks. Suppose that $(s, w) \models \langle a\rangle\mathtt{tt}$. Then $(s, w) \models x \ \underline{\mathtt{in}}\ \mathbb{\forall}_{\{a\}}(x = 0)$.*
2. *Let $\langle n, u\rangle$ be a state of a timed automaton, and $w$ be a valuation for the formula clocks. Suppose that $(\langle n, u\rangle, w) \models x \ \underline{\mathtt{in}}\ \mathbb{\forall}_{\{a\}}(x = 0)$. Then $(\langle n, u\rangle, w) \models \langle a\rangle\mathtt{tt}$.*

*Thus the formula $\langle a\rangle\mathtt{tt}$ ($a \in \mathcal{U}$) is equivalent to $x \ \underline{\mathtt{in}}\ \mathbb{\forall}_{\{a\}}(x = 0)$ over states of timed automata.*

*Proof.* The equivalence of the formulae $\langle a\rangle\mathtt{tt}$ and $x \ \underline{\mathtt{in}}\ \mathbb{\forall}_{\{a\}}(x = 0)$ over states of timed automata is an immediate consequence of statements 1 and 2 of the lemma. These we now prove separately.

1. Let $s$ be a state of a TLTS, and $w$ be a valuation for the formula clocks. Suppose that $(s,w) \models \langle a \rangle \mathtt{tt}$. We show that $(s,w) \models x \ \underline{\mathtt{in}}\ \forall\!\!\!\forall_{\{a\}}(x=0)$. To this end, assume that

$$s \xrightarrow{\tau}^* s' \overset{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'' \quad .$$

   We wish to argue that $(s'', [x \to d](w+d)) \models x = 0$, *i.e.*, that $d = 0$.
   First of all, note that $(s', w) \models \langle a \rangle \mathtt{tt}$ because $(s,w) \models \langle a \rangle \mathtt{tt}$ and $s \xrightarrow{\tau}^* s'$ (Proposition 4.6). A simple induction on the length of the derivation $s' \overset{\epsilon(d)}{\Longrightarrow}_{\{a\}} s''$ now shows that $d = 0$, which was to be shown.
2. We prove the contrapositive statement. To this end, assume that, for some state of a timed automaton $\langle n, u \rangle$ and valuation $w$ for the formula clocks, $(\langle n,u \rangle, w) \not\models \langle a \rangle \mathtt{tt}$. We show that $(\langle n,u \rangle, w) \not\models x \ \underline{\mathtt{in}}\ \forall\!\!\!\forall_{\{a\}}(x=0)$.
   Since $(\langle n,u \rangle, w) \not\models \langle a \rangle \mathtt{tt}$, there exists a state $\langle n', u' \rangle$ such that $\langle n,u \rangle \xrightarrow{\tau}^* \langle n', u' \rangle \stackrel{a}{\not\longrightarrow}$. We show that $(\langle n', u' \rangle, [x \to 0]w) \not\models \forall\!\!\!\forall_{\{a\}}(x=0)$. To this end, it is sufficient to find a positive real number $d$ such that $\langle n', u' \rangle \xrightarrow{\epsilon(d)}_{\{a\}} \langle n', u'+d \rangle$. As $\langle n', u' \rangle \stackrel{a}{\not\longrightarrow}$ and $a$ is urgent, backward persistence of urgent actions (see Definition 2.1 to follow) yields that $\langle n', u'+d \rangle \stackrel{a}{\not\longrightarrow}$ for every $d \geq 0$. Thus, for example, it holds that $\langle n', u' \rangle \xrightarrow{\epsilon(1)}_{\{a\}} \langle n', u'+1 \rangle$. Since $(\langle n', u'+1 \rangle, [x \to 1]w) \not\models x = 0$, the claim follows.

The proof of the lemma is now complete. □

Using Lemma 4.11, it is now a simple matter to show that the property language $L^-_{\forall S}$ is at least as expressive as $L_{\forall S}$ over states of timed automata. In fact, every formula in $L_{\forall S}$ can be translated into an equivalent one (over states of timed automata) in $L^-_{\forall S}$ simply by replacing every occurrence of formulae of the form $\langle a \rangle \mathtt{tt}$ with $x \ \underline{\mathtt{in}}\ \forall\!\!\!\forall_{\{a\}}(x=0)$. It follows from this analysis that $L_{\forall S}$ and $L^-_{\forall S}$ are indeed equally expressive over states of timed automata.

Note that Lemma 4.11(2) does *not* hold over models given by states of arbitrary TLTSs. Consider, for instance, the TLTS consisting of one state $s$ and of the (delay) transition $s \xrightarrow{\epsilon(0)} s$. Then, for every valuation $w$ of the formula clocks, $(s,w) \models x \ \underline{\mathtt{in}}\ \forall\!\!\!\forall_{\{a\}}(x=0)$. On the other hand, $s$ affords no $a$-labelled transition, and therefore $(s,w) \not\models \langle a \rangle \mathtt{tt}$. (The reader can easily verify that such a state $s$ cannot be the state of a timed automaton.) Thus the encoding of the basic proposition $\langle a \rangle \mathtt{tt}$ into $L^-_{\forall S}$ which is appropriate over the class of models given by states of timed automata does not extend to the larger class of models given by states of TLTSs. We shall now show that there is *no* formula in $L^-_{\forall S}$ that is equivalent to $\langle a \rangle \mathtt{tt}$ over states of TLTSs, and thus that the language $L_{\forall S}$ is strictly more expressive than $L^-_{\forall S}$ over such models. To this end, we shall exhibit two states $s$ and $t$ of a TLTS with the following properties:

– for every valuation $w$ of the formula clocks, $(t,w)$ satisfies $\langle a \rangle \mathtt{tt}$, but $(s,w)$ does not and
– for every valuation $w$ of the formula clocks, the collection of formulae in $L^-_{\forall S}$ which are satisfied by $(t,w)$ is included in that of the formulae satisfied by $(s,w)$.

These two properties together ensure that there is no formula in $L^-_{\forall S}$ that is equivalent to $\langle a \rangle \mathtt{tt}$ over states of TLTSs.

Consider the states $s$ and $t$ of a TLTS whose transitions are as follows:

$$s \xrightarrow{\epsilon(0)} s$$
$$t \xrightarrow{\epsilon(0)} t$$
$$t \xrightarrow{a} t$$

Obviously, independently of the valuation $w$ for the formula clocks, the extended state $(t,w)$ satisfies $\langle a \rangle \mathtt{tt}$, but, as we have remarked above, $(s,w)$ does not. On the other hand, every formula in $L^-_{\forall S}$ which is satisfied by $(t,w)$ is also satisfied by $(s,w)$.

**Lemma 4.12.** *For every $\varphi \in L^-_{\forall S}$ and valuation $w$ for the formula clocks, if $(t, w) \models \varphi$, then $(s, w) \models \varphi$.*

*Proof.* It is sufficient to show that the relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{((s, w), \varphi) \mid \varphi \in L^-_{\forall S} \text{ and } (t, w) \models \varphi\}$$

is a satisfiability relation. We leave the routine details of such a proof to the reader, and only present the case $\varphi \equiv \mathbb{\forall}_S \phi$.

Assume that $((s, w), \mathbb{\forall}_S \phi) \in \mathcal{R}$, and that $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$ for some non-negative real number $d$ and state $s'$. We wish to argue that $((s', w + d), \phi) \in \mathcal{R}$. To this end, note that $d = 0$ and $s' = s$. Moreover, $t$ affords the transition $t \stackrel{\epsilon(0)}{\longrightarrow}_S t$ regardless of whether $a \in S$ or not. Since $(t, w) \models \mathbb{\forall}_S \phi$, we infer that $(t, w) \models \phi$. Finally, by the definition of $\mathcal{R}$, it follows that $((s, w), \phi) \in \mathcal{R}$, which was to be shown. $\square$

In light of Lemma 4.12 and of the previous considerations, we may finally infer that $L_{\forall S}$ is strictly more expressive than $L^-_{\forall S}$ over states of TLTSs, and this completes the proof of Theorem 4.10. As we shall see in what follows, this increase in expressive power is, however, achieved by adding a basic proposition, *viz.* $\langle a \rangle \mathtt{tt}$, to $L^-_{\forall S}$ which cannot be tested over the class of models given by states of TLTSs. (see Proposition 4.17.)

We now proceed to study the relative expressive power of the languages SBLL and $L^-_{\forall S}$. We shall show that SBLL is strictly less expressive than $L^-_{\forall S}$ over states of timed automata, and *a fortiori* over states of arbitrary TLTSs.

The difference between the property language introduced in Definition 4.4 and the one considered in [ABL98] (see also Definition 4.1) is that, unlike the language SBLL studied *ibidem*, $L_{\forall S}$ has universal delay constructs which are parameterized by a set of urgent actions $S$. Indeed, the property language SBLL is just the sublanguage of $L_{\forall S}$ in which only occurrences of $\mathbb{\forall}_\emptyset$ are allowed. We shall now show that the use of $\mathbb{\forall}_S$ for arbitrary sets of urgent actions $S$ leads to a more expressive property language over states of timed automata. This is the import of the following result:

**Theorem 4.13.** *The property language $L^-_{\forall S}$ is strictly more expressive than* SBLL *over states of timed automata.*

To establish the above claim, we shall prove that

1. every formula in SBLL has an equivalent one (over states of timed automata) in $L^-_{\forall S}$, and that
2. there is a formula in $L^-_{\forall S}$ that is not equivalent over states of timed automata to any formula in SBLL.

The first of these statements is easily seen to hold because there is a syntactic embedding of SBLL into $L_{\forall S}$ which simply maps every occurrence of $\mathbb{\forall}$ to $\mathbb{\forall}_\emptyset$, and every occurrence of basic propositions of the form $\langle a \rangle \mathtt{tt}$ to the formula $x \ \underline{\mathtt{in}}\ \mathbb{\forall}_{\{a\}}(x = 0)$. In light of Lemma 4.11, such an embedding clearly preserves the semantics of formulae because the transition relations $\stackrel{\epsilon(d)}{\Longrightarrow}$ and $\stackrel{\epsilon(d)}{\Longrightarrow}_\emptyset$ coincide. To establish the second of the claims, it is sufficient to exhibit two timed automata $A$ and $B$, and a formula $\phi$ in $L^-_{\forall S}$ such that

- every formula in SBLL that is satisfied by $A$ is also satisfied by $B$, and
- $A$ satisfies $\phi$, but $B$ does not.

Let $A$ and $B$ be the timed automata whose initial states are $\langle t_0, [x \to 0] \rangle$ and $\langle s_0, [x \to 0] \rangle$ in Figure 6, respectively. Take $\phi \equiv \mathbb{\forall}_{\{a\}}[b]\mathtt{ff}$. We have already seen that $A$ satisfies $\phi$, but $B$ does not (see Example 4.9), and thus that the behaviour of these two automata can be differentiated using a property in the language $L^-_{\forall S}$. We shall now show that every formula in SBLL that is satisfied by $A$ is also satisfied by $B$. This is the import of the following result:

**Lemma 4.14.** *Let $t_0$ and $s_0$ be as in Figure 6. Then, for every formula $\varphi \in$ SBLL and valuation $v$ for the clocks in $K$,*

$$((\langle t_0, [x \to 0]\rangle, v) \models \varphi \textit{ implies } ((\langle s_0, [x \to 0]\rangle, v) \models \varphi \ .$$

*Proof.* The proof of this lemma can be found in Appendix A. □

The above analysis yields that the property $\forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$ cannot be expressed in SBLL over the class of models given by states of timed automata. Assume, in fact, towards a contradiction, that there is a formula $\phi$ in SBLL which is equivalent to $\forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$ over states of timed automata. Then $A$ satisfies $\phi$, but $B$ does not (Example 4.9). However, this contradicts Lemma 4.14. Hence no formula in SBLL can be equivalent to $\forall\!\!\!\forall_{\{a\}}[b]\mathtt{ff}$ over states of timed automata. This completes the proof of Theorem 4.13.

### 4.7 Testing $L^{-}_{\forall S}$

In Section 4.2 we have seen how we can perform tests for the property language SBLL. We now aim at extending our result on SBLL to the property language $L^{-}_{\forall S}$.

**Theorem 4.15.** *For every closed formula $\varphi$ in $L^{-}_{\forall S}$, there exists a test automaton $T_\varphi$, over the set of clocks $\{k\} \cup \mathtt{clocks}(\varphi)$, where $k$ does not occur in $\mathtt{clocks}(\varphi)$, that tests for it.*

*Proof.* We just have to complete the proof of Theorem 4.2 by considering the new operator $\forall\!\!\!\forall_S\varphi$. The proof for SBLL was based on a structural induction on formulae (that are not necessarily closed). We thus assume that we can construct a test automaton for the property $\varphi$. We will then construct a test automaton for the property $\forall\!\!\!\forall_S\varphi$. Thus let us consider the test automaton $T_{\forall_S\varphi}$ depicted on Figure 7. Assume that a reject node of $T_{\forall_S\varphi}$ is reachable in the parallel composition

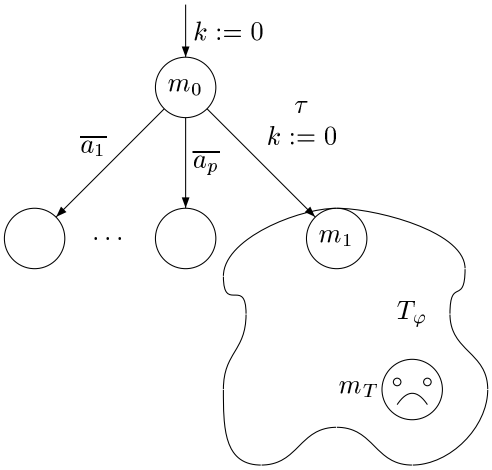



**Figure 7.** $T_{\forall_S\varphi}$ if $S = \{a_1, \dots, a_p\}$

of some transition system $\mathcal{T}$ and $T_{\forall_S\varphi}$. Then, it means that the following computation is allowed in $(\mathcal{T} \parallel T_{\forall_S\varphi})\backslash\mathsf{Act}$ ($d \geq 0$ is some real):

$$\begin{aligned}
(s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} &\stackrel{\epsilon(d)}{\Longrightarrow} (s_1 \parallel \langle m_0, ([k \to 0]w_0) + d\rangle)\backslash\mathsf{Act} \\
&\stackrel{\tau}{\longrightarrow} (s_1 \parallel \langle m_1, [k \to 0](w_0 + d)\rangle)\backslash\mathsf{Act} \\
&\leadsto^* (s_2 \parallel \langle m_T, w\rangle)\backslash\mathsf{Act} \ .
\end{aligned}$$

Moreover, the first $\stackrel{\epsilon(d)}{\Longrightarrow}$ transition is in fact a $\stackrel{\epsilon(d)}{\Longrightarrow}_S$ transition because there is no allowed synchronization between an $a_i$ and an $\overline{a_i}$ action. It follows that $(s_1, ([k \to 0]w_0) + d) \not\models \varphi$, and thus that $(s_0, [k \to 0]w_0) \not\models \mathbb{\forall}_S\varphi$.

Conversely, let us assume that $(s_0, w_0) \not\models \mathbb{\forall}_S\varphi$. It means that there exists some real $d \geq 0$ and some state $s_1$ of $\mathcal{T}$ such that $s_0 \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1$ and $(s_1, w_0 + d) \not\models \varphi$. Thus, in the parallel composition of $\mathcal{T}$ and $T_{\mathbb{\forall}_S\varphi}$, the following computation is allowed:

$$\begin{array}{rll} (s_0 \parallel \langle m_0, [k \to 0]w_0\rangle)\backslash\mathsf{Act} & \stackrel{\epsilon(d)}{\Longrightarrow}_S & (s_1 \parallel \langle m_0, ([k \to 0]w_0) + d\rangle)\backslash\mathsf{Act} \\ & \stackrel{\tau}{\longrightarrow} & (s_1 \parallel \langle m_1, [k \to 0](w_0 + d)\rangle)\backslash\mathsf{Act} \\ & \leadsto^* & (s \parallel \langle m_T, w_T\rangle)\backslash\mathsf{Act} \ . \end{array}$$

Thus, a reject node of $T_{\mathbb{\forall}_S\varphi}$ is reachable in $(\mathcal{T} \parallel T_{\mathbb{\forall}_S\varphi})\backslash\mathsf{Act}$. □

As a consequence of the above theorem and of Lemma 4.11, we obtain the following stronger result for timed automata.

**Corollary 4.16.** *Let $\varphi$ be a closed formula in $L_{\forall S}$. Then there exists a test automaton $T_\varphi$, over the set of clocks $\{k\} \cup \mathtt{clocks}(\varphi)$, that tests for it over states of timed automata.*

As remarked in Section 4, the property language $L^-_{\forall S}$ only allows for a restricted use of the 'or' operator. Moreover, even though formulae of the form $\langle a\rangle\mathtt{tt}$ can be encoded in $L^-_{\forall S}$ over states of timed automata (Lemma 4.11) when $a$ is an urgent action, it is not possible to express $\langle a\rangle\mathtt{tt}$ within $L^-_{\forall S}$ if $a$ is *not* urgent. These restrictions are justified by the following negative results, where we consider extensions of the language $L^-_{\forall S}$ with the above-mentioned operator and atomic formula.

**Proposition 4.17.**

1. *The formula $\langle a\rangle\mathtt{tt}$ is not testable over states of TLTSs, regardless of whether $a$ is urgent or not.*
2. *The formula $\langle a\rangle\mathtt{tt}$ ($a$ not urgent) is not testable over states of timed automata.*
3. *The formula $[a]\mathtt{ff} \vee [b]\mathtt{ff}$ ($a, b$ not urgent) is not testable over states of timed automata.*

*Proof.* The proof of this proposition is presented in Appendix A. □

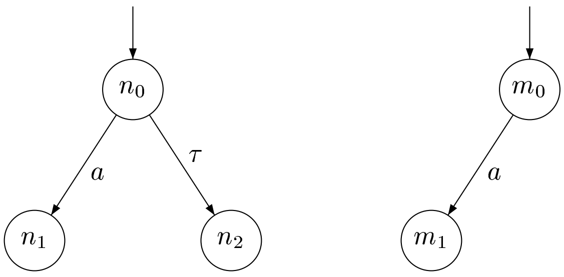



**Figure 8.** The test automata used in the proof of Proposition 4.17(2)

Statement 1 in the above proposition justifies our previous claim to the effect that the added expressive power of the language $L_{\forall S}$ with respect to $L^-_{\forall S}$ over states of arbitrary TLTSs comes from the addition of a construct, *viz.* $\langle a\rangle\mathtt{tt}$, which is not testable over states of TLTSs.

In the following section, we shall address the problem of the expressive completeness of $L^-_{\forall S}$ with respect to reachability testing. In particular, we shall establish our main result, *viz.* that the properties expressible in $L^-_{\forall S}$ are precisely those that are testable using reachability analysis over test automata.

## 5 Compositionality and Expressive Completeness

We have previously shown that every property $\varphi$ which can be expressed in the language $L_{\forall S}$ (and, *a fortiori*, in SBLL) is testable over states of timed automata, and that $L^{-}_{\forall S}$ is testable over states of TLTSs, in the sense of Definition 3.4. We now address the problem of the expressive completeness of these property languages with respect to test automata and (reachability) testing. More precisely, we study whether all properties that are testable over TLTSs can be expressed in the property languages SBLL and $L^{-}_{\forall S}$ — in the sense that, for every test automaton $T$, there exists a formula $\psi_T$ such that every extended state of a TLTS passes the test $T$ if and only if it satisfies $\psi_T$. Indeed, we have already enough information to claim that the language SBLL is strictly less expressive than the formalism of test automata. In fact, the automaton depicted in Figure 5 is nothing but the test automaton for the formula $\mathbb{V}_{\{a\}}[b]\mathtt{ff}$, which cannot be expressed in SBLL (see the proof of Theorem 4.13). Our aim in this section is to argue that, unlike SBLL, the language $L^{-}_{\forall S}$ *is* expressive complete, in the sense that every test automaton $T$ may be expressed as a property in the language $L^{-}_{\forall S}$ in the precise technical sense outlined above. This amounts to establishing a kind of expressive completeness result for our specification language $L^{-}_{\forall S}$, with respect to reachability questions over the formalism of test automata, akin to classic ones presented in, *e.g.*, [Kle56, GPSS80]. In the proof of this expressive completeness result, we shall follow an indirect approach by focusing on the compositionality of a property language $\mathbb{L}$ with respect to test automata and the parallel composition operator $\|$. As we shall see (see Proposition 5.3), if a property language $\mathbb{L}$ is compositional with respect to timed automata and $\|$ (see Definition 5.2) then it is complete with respect to test automata and reachability testing (see Definition 5.1). We show that SBLL is *not* compositional with respect to timed automata and $\|$, but its variation $L^{-}_{\forall S}$ considered in this paper is.

We begin with some preliminary definitions, introducing the key concepts of compositionality and (expressive) completeness.

**Definition 5.1 (Expressive completeness with respect to test automata and testing).** Let $\mathbb{L}$ be a property language over the set of clocks $K$. We say that $\mathbb{L}$ is *(expressive) complete* (with respect to test automata and testing) if for every test automaton $T$ there exists a formula $\varphi_T \in \mathbb{L}$ such that,

for every extended state $(s,u)$ of a TLTS, $(s,u) \models \varphi_T$ iff $(s,u)$ passes the test $T$.

Compositionality, on the other hand, is formally defined as follows:

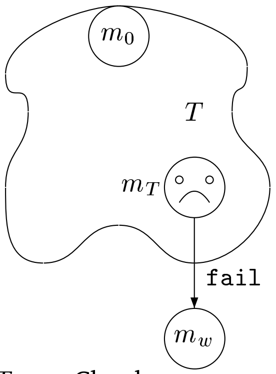
**Definition 5.2 (Compositionality with respect to test automata and $\|$).** For a property language $\mathbb{L}$ over the set of clocks $K$, we say that $\mathbb{L}$ is *compositional* (with respect to test automata and $\|$) if, for every $\varphi \in \mathbb{L}$ and every test automaton $T = \langle \mathsf{Act}_\tau, N, N_T, n_0, C, C_0, E\rangle$ (with $C$ disjoint from $\mathtt{clocks}(\varphi)$), there exists a formula $\varphi/T \in \mathbb{L}$ over the set of clocks $\mathtt{clocks}(\varphi) \cup C$ such that, for every state $s$ of a TLTS and every valuation $u$ for $K$,

$$(s \parallel (\langle n_0, [C_0 \to 0](u \restriction C)\rangle, u)) \models \varphi \;\Leftrightarrow\; (s, [C_0 \to 0]u) \models \varphi/T \ .$$

Our interest in compositionality stems from the following result that links it to the notion of completeness. In the sequel, we use $\mathbb{L}_{bad}$ to denote the property language that only consists of the formula $\mathbb{V}_{\emptyset}[\mathtt{fail}]\mathtt{ff}$, where $\mathtt{fail}$ is a fresh action not contained in Act.

**Proposition 5.3.** *Let $\mathbb{L}$ be a property language (over a set of clocks $K$) that includes $\mathbb{L}_{bad}$. Suppose that $\mathbb{L}$ is compositional with respect to test automata and the parallel composition operator $\|$. Then $\mathbb{L}$ is complete with respect to test automata and testing.*

*Proof.* Assume that the formula $\mathbb{V}_{\emptyset}[\mathtt{fail}]\mathtt{ff} \in \mathbb{L}$ and $\mathbb{L}$ is compositional with respect to test automata and $\|$. We show that $\mathbb{L}$ is complete. To this end, let $T = \langle \mathsf{Act}_\tau, N, N_T, m_0, C, C_0, E\rangle$ be a test automaton. We wish to argue that there exists a formula $\varphi_T \in \mathbb{L}$ meeting the requirements in Definition 5.1. To this end, we begin by extending $T$ with a new node $m_w$ and edges $e = \langle m_T, m_w, \mathtt{fail}, \emptyset, \mathtt{tt}\rangle$, where $m_T$ is a reject node of $T$, and $\mathtt{fail}$ is an action not in Act, as depicted below:




Call the resulting test automaton $T_{\mathtt{fail}}$. Clearly every extended state $(s,u)$ of a TLTS passes the test $T$ iff it passes the test $T_{\mathtt{fail}}$. As $\mathbb{L}$ is compositional, we may define $\varphi_T$ to be the formula $C_0 \ \underline{\mathtt{in}}\ ((\forall\!\!\!\forall_{\emptyset}[\mathtt{fail}]\mathbf{ff})/T_{\mathtt{fail}})$ given by Definition 5.2. We now proceed to argue that such a formula $\varphi_T$ does indeed meet the requirements in Definition 5.1.

Let $(s,v)$ be an extended state of an arbitrary TLTS. Then we reason as follows:

$$
\begin{array}{lll}
(s,v) \text{ passes the test T} & \text{iff} & (s,v) \text{ passes the test } T_{\mathtt{fail}} \\
 & & \quad \text{(By the construction of } T_{\mathtt{fail}}) \\
 & \text{iff} & (s \parallel \langle m_0, [C_0 \to 0](v \restriction C)\rangle)\backslash \mathsf{Act} \text{ cannot reach a reject node in } T_{\mathtt{fail}} \\
 & & \quad \text{(By the definition of passing a test)} \\
 & \text{iff} & (s \parallel \langle m_0, [C_0 \to 0](v \restriction C)\rangle)\backslash \mathsf{Act} \models \forall\!\!\!\forall_{\emptyset}[\mathtt{fail}]\mathbf{ff} \\
 & & \quad \text{(By the construction of } T_{\mathtt{fail}}) \\
 & \text{iff} & (s \parallel \langle m_0, [C_0 \to 0](v \restriction C)\rangle) \models \forall\!\!\!\forall_{\emptyset}[\mathtt{fail}]\mathbf{ff} \\
 & & \quad \text{(As } \mathtt{fail} \text{ is not contained in } \mathsf{Act}) \\
 & \text{iff} & (s, [C_0 \to 0]v) \models (\forall\!\!\!\forall_{\emptyset}\mathtt{fail}]\mathbf{ff})/T_{\mathtt{fail}} \\
 & & \quad \text{(As } \mathbb{L} \text{ is compositional and } \mathtt{clocks}((\forall\!\!\!\forall_{\emptyset}[\mathtt{fail}]\mathbf{ff})/T_{\mathtt{fail}}) \subseteq C) \\
 & \text{iff} & (s,v) \models C_0 \ \underline{\mathtt{in}}\ ((\forall\!\!\!\forall_{\emptyset}[\mathtt{fail}]\mathbf{ff})/T_{\mathtt{fail}}) \\
 & & \quad \text{(As } \models \text{ is the largest satisfiability relation)} \\
 & \text{iff} & (s,v) \models \varphi_T \ .
\end{array}
$$

Since $T$ was an arbitrary test automaton, we can conclude that the property language $\mathbb{L}$ is complete with respect to test automata and testing. □

Since both SBLL and $L^{-}_{\forall S}$ are extensions of $\mathbb{L}_{bad}$, in light of the above proposition an approach to proving that they are expressive complete is to establish that they are compositional with respect to test automata and $\parallel$. However, we begin by arguing that such an approach is doomed to fail for SBLL.

**Proposition 5.4.** *The property language* SBLL *is not compositional with respect to test automata and* $\parallel$.

*Proof.* Assume, towards a contradiction, that SBLL is compositional with respect to $\parallel$. Since SBLL obviously includes $\mathbb{L}_{bad}$, Proposition 5.3 yields that SBLL is complete with respect to test automata and testing. Hence, in particular, there is a formula $\varphi \in$ SBLL such that, for every extended state $(s,u)$,

$$(s,u) \text{ passes the test } T_\phi \text{ iff } (s,u) \models \varphi$$

where $T_\phi$ is the test automaton for the formula $\phi \equiv \forall\!\!\!\forall_{\{a\}}[b]\mathbf{ff}$ depicted in Figure 5. Let $t_0$ and $s_0$ be as in Figure 6. In light of Theorem 4.15 and Example 4.9, for every valuation $w$ of the clocks in $K$, $(\langle t_0, [x \to 0]\rangle, w)$ satisfies $\varphi$ but $(\langle s_0, [x \to 0]\rangle, w)$ does not. However, since $\varphi$ is contained in SBLL, this contradicts Lemma 4.14. □

On the other hand, as we shall now show, $L^{-}_{\forall S}$ is compositional with respect to test automata and $\parallel$, and thus expressive complete with respect to test automata and testing.

We begin by defining a quotient construction for formulae of $L_{\forall S}$, in the spirit of those given for different property languages and over different models in, *e.g.*, [LX91, And95, LL95].

**Definition 5.5 (Quotient construct for $L_{\forall S}$).** Let $A$ be a timed automaton, and $n$ one of its nodes. Let $\varphi$ be a formula in $L^{-}_{\forall S}$. We define the formula $\varphi/n$ (read '$\varphi$ quotiented by $n$') as shown in Table 5.

One remark about the definition presented in Table 5 is now in order. The definition of the quotient formula $\varphi/n$ presented *ibidem* should be read as yielding a finite list of recursion equations, over variables of the form $\phi/m$, for every formula $\varphi$ and node $n$ of a timed automaton. The quotient formula $\varphi/n$ itself is the component associated with $\varphi/n$ in the largest solution of the system of equations having $\varphi/n$ as leading variable. For instance, if $\varphi$ is the formula $[a]\mathtt{ff}$ and $n$ is a node of a timed automaton whose only edge is $\langle n, n, \bar{b}, \emptyset, \mathtt{tt}\rangle$, then, as the reader can easily verify, $\varphi/n$ is the largest solution of the recursion equation:

$$\varphi/n \stackrel{\text{def}}{=} [a]\mathtt{ff} \wedge [b](\varphi/n)$$

which corresponds to the formula $\mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$ in the property language $L^{-}_{\forall S}$. This formula states the, intuitively clear, fact that the state $(s \parallel n)$ cannot perform a $\stackrel{a}{\Longrightarrow}$-transition iff $s$ cannot execute such a step no matter how it engages in a sequence of synchronizations on $b$ with $n$.

Note that, as in the previously discussed case, the quotient of a recursion-free formula may therefore be a formula involving recursion. It can be shown that this is inevitable, because the recursion-free fragment of $L^{-}_{\forall S}$ is not compositional with respect to test automata and $\parallel$.

We are now in a position to state the following key result.

**Theorem 5.6.** *Let $\varphi$ be a closed formula in $L^{-}_{\forall S}$. Suppose that $s$ is a state of a TLTS, and $n'$ is a state of a timed automaton over set of clocks $C$. Suppose furthermore that $u$ and $v'$ are valuations for the disjoint set of clocks $\mathtt{clocks}(\varphi)$ and $C$, respectively. Then*

$$(s \parallel \langle n', v'\rangle, u) \models \varphi \iff (s, v' : u) \models \varphi/n' \ .$$

*Proof.* See Appendix B for the complete proof of this theorem. □

**Corollary 5.7.** *The property language $L^{-}_{\forall S}$ is compositional with respect to test automata and the parallel composition operator $\parallel$.*

**Theorem 5.8.** *The property language $L^{-}_{\forall S}$ is complete with respect to test automata and testing.*

*Proof.* By Proposition 5.3 and the previous corollary. □

**Theorem 5.9.** *The property language $L^{-}_{\forall S}$ is the least expressive extension of $\mathbb{L}_{bad}$ that is compositional with respect to test automata and $\parallel$.*

*Proof.* Assume that $\mathbb{L}$ is a property language that extends $\mathbb{L}_{bad}$ and is compositional with respect to timed automata and testing. We show that every property in $L_{\forall S}$ is logically equivalent to one in $\mathbb{L}$, i.e., that $\mathbb{L}$ is at least as expressive as $L_{\forall S}$. To this end, let $\varphi$ be a property in $L_{\forall S}$. By Theorem 4.15, there is a test automaton $T_\varphi$ such that, for every extended state $(s, u)$,

$$(s, u) \models \varphi \text{ iff } (s, u) \text{ passes the test } T_\varphi.$$

Since $\mathbb{L}$ is an extension of $\mathbb{L}_{bad}$ that is compositional with respect to timed automata and testing, Proposition 5.3 yields that $\mathbb{L}$ is complete. Thus there is a formula $\phi \in \mathbb{L}$ such that, for every extended state $(s, u)$,

$$(s, u) \models \phi \text{ iff } (s, u) \text{ passes the test } T_\varphi.$$

It follows that $\phi$ and $\varphi$ are satisfied by precisely the same extended states, and are therefore equivalent over states of TLTSs. □

$$
\begin{aligned}
\mathtt{ff}/n' &\stackrel{\text{def}}{=} \mathtt{ff} \\[1em]
(\phi_1 \wedge \phi_2)/n' &\stackrel{\text{def}}{=} \phi_1/n' \wedge \phi_2/n' \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\phi_1 \wedge \phi_2)/n'_e) \\
&\quad \wedge \bigwedge_{b \in \mathsf{Act}} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\phi_1 \wedge \phi_2)/n'_e)) \\
(g \vee \phi)/n' &\stackrel{\text{def}}{=} (g \vee (\phi/n')) \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (g \vee \phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in \mathsf{Act}} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (g \vee \phi)/n'_e)) \\
([a]\phi)/n' &\stackrel{\text{def}}{=} [a](\phi/n') \\
&\quad \wedge \bigwedge_{e \in E(n',a)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi/n'_e) \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ ([a]\phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in Act} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ ([a]\phi)/n'_e)) \\
(x \ \underline{\mathtt{in}}\ \phi)/n' &\stackrel{\text{def}}{=} x \ \underline{\mathtt{in}}\ (\phi/n') \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (x \ \underline{\mathtt{in}}\ \phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in Act} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (x \ \underline{\mathtt{in}}\ \phi)/n'_e)) \\
(\forall\!\!\!\forall_S \phi)/n' &\stackrel{\text{def}}{=} \forall\!\!\!\forall_{S \cup \overline{\mathcal{U}(n')}}\ (\phi/n' \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\forall\!\!\!\forall_S \phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in \mathsf{Act}} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\forall\!\!\!\forall_S \phi)/n'_e))) \qquad \text{if } S \cap \mathcal{U}(n') = \emptyset \\
(\forall\!\!\!\forall_S \phi)/n' &\stackrel{\text{def}}{=} (\phi/n' \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\forall\!\!\!\forall_S \phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in \mathsf{Act}} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\forall\!\!\!\forall_S \phi)/n'_e))) \qquad \text{if } S \cap \mathcal{U}(n') \neq \emptyset \\
X/n' &\stackrel{\text{def}}{=} X \\
\mathtt{max}(X,\phi)/n' &\stackrel{\text{def}}{=} (\phi\{\mathtt{max}(X,\phi)/X\})/n' \\
&\quad \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \mathtt{max}(X,\phi)/n'_e) \\
&\quad \wedge \bigwedge_{b \in Act} \ \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\mathtt{max}(X,\phi))/n'_e))
\end{aligned}
$$

**Table 5.** Quotient construct for $L^{-}_{\forall S}$

## 6 Property Languages *vs* Behavioural Preorders

In the verification of realistic reactive systems, it is often useful to replace the individual components of the system under verification with more abstract versions before building the model of the complete system. This abstraction must, of course, be carried out in such a way that every property enjoyed by the resulting abstract model should also hold of the original, more detailed system description, and can be conveniently justified by means of behavioural characterizations of the equivalence/preorder induced by the property language under consideration. (See, *e.g.*, [TB97] for an impressive recent example of this general strategy applied to the verification of a high bandwidth communication chip.) In this section, we shall provide a behavioural characterization of the preorder on states induced by the property language $L_{\forall S}$ over a subclass of timed automata. A key step in such a characterization is to show how the property language $L_{\forall S}$ can be used to define characteristic properties [SI94] for nodes of $\tau$-free, deterministic timed automata with respect to a timed version of the ready simulation preorder (also known as 2/3-bisimulation) behavioural [LS91, BIM95]. As $\tau$-free, deterministic timed automata are prime candidates for use as abstractions of more complex systems, the use of characteristic formulae allows us to formally, and automatically, justify abstractions using the model-checking algorithm via reachability testing we have presented in [ABL98], and implemented in UPPAAL. The timed version of the ready simulation preorder that we shall consider is defined as follows:

**Definition 6.1.** Let $\mathcal{T} = \langle \mathcal{S}, \mathcal{L}, s^0, \longrightarrow \rangle$ be a TLTS. We define the preorder $\preceq_w$ as the largest binary relation over $\mathcal{S}$ such that if $s_1 \preceq_w s_2$, then

1. whenever $s_1 \stackrel{a}{\Longrightarrow} s_1'$, then $s_2 \stackrel{a}{\Longrightarrow} s_2'$ for some $s_2'$ such that $s_1' \preceq_w s_2'$;
2. whenever $s_1 \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1'$, then $s_2 \stackrel{\epsilon(d)}{\Longrightarrow}_S s_2'$ for some $s_2'$ such that $s_1' \preceq_w s_2'$; and
3. if $a \in \mathcal{U}$ and $s_2 \stackrel{a}{\longrightarrow} s_2'$ for some $s_2'$, then $s_1 \stackrel{a}{\longrightarrow} s_1'$ for some $s_1'$.

For timed automata $A$ and $B$, we write $A \preceq_w B$ iff $\langle n_0, u_0 \rangle \preceq_w \langle m_0, v_0 \rangle$, where $\langle n_0, u_0 \rangle$ and $\langle m_0, v_0 \rangle$ are the initial states of $A$ and $B$, respectively.
*Timed simulation* is the largest relation over $\mathcal{S}$ satisfying clauses 1–2 above.

The intuition captured by $\preceq_w$ is that if $s_1 \preceq_w s_2$, then $s_2$ offers a possibly "more abstract" version of the behaviour of $s_1$. It is well-known that $\preceq_w$ and timed simulation are preorders over $\mathcal{S}$.

The main usage that we envisage for the relation $\preceq_w$ is in justifying abstraction steps in verification. To this end, we expect that if $s_1 \preceq_w s_2$ holds, then every property of the abstract state $s_2$ expressible in $L_{\forall S}$ is also a property of $s_1$. This is the import of the following result.

**Theorem 6.2.** *Assume that $s_1 \preceq_w s_2$. Then, for every $\varphi \in L_{\forall S}$ and valuation $u$ for the formula clocks, $(s_2, u) \models \varphi$ implies $(s_1, u) \models \varphi$.*

*Proof.* Consider the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} = \{((s,u),\varphi) \mid \varphi \in L_{\forall S},\ (t,u) \models \varphi \text{ and } \preceq_w t,\ \text{for some } t\} \ .$$

We prove that $\mathcal{R}$ satisfies the implications in Table 4. Assume, to this end, that $(\langle s,u \rangle, \varphi) \in \mathcal{R}$ — *i.e.*, that there exists a state $t$ such that $\langle t,u \rangle \models \varphi$ and $s \preceq_w t$. We proceed by a case analysis on the form that $\varphi$ may take, and only present a few paradigmatic cases in such a proof, as the remaining ones follow a similar pattern.

- CASE $\varphi \equiv g \vee \phi$. Assume that $s \stackrel{\tau}{\longrightarrow}^* s'$. We prove that either $g(u)$ holds or $((s',u),\phi) \in \mathcal{R}$. To this end, suppose that $g(u)$ does not hold. Then, as the relation $\stackrel{\tau}{\longrightarrow}^*$ coincides with $\stackrel{\epsilon(0)}{\Longrightarrow}_\emptyset$, there exists a state $t'$ such that $t \stackrel{\tau}{\longrightarrow}^* t'$ and $s' \preceq_w t'$. Moreover, as $(t,u)$ satisfies $\phi$ (because $(t,u) \models \varphi$ and $g(u)$ does not hold), so does $(t',u)$ (Proposition 4.6). By the definition of $\mathcal{R}$, it follows that $((s',u),\phi) \in \mathcal{R}$, which was to be shown.

– CASE $\varphi \equiv [a]\phi$. Assume that $s \stackrel{a}{\Longrightarrow} s'$. We prove that $((s', u), \phi) \in \mathcal{R}$. To this end, note that, as $s \preceq_w t$ and $s \stackrel{a}{\Longrightarrow} s'$, there is a state $t'$ such that $t \stackrel{a}{\Longrightarrow} t'$ and $s' \preceq_w t'$. As $(t, u)$ satisfies $[a]\phi$, it follows that $(t', u)$ satisfies $\phi$. By the definition of $\mathcal{R}$, we conclude that $((s', u), \phi) \in \mathcal{R}$, which was to be shown.
– CASE $\varphi \equiv \langle a \rangle \mathtt{tt}$ for some $a \in \mathcal{U}$. Assume that $s \stackrel{\tau}{\longrightarrow}^* s'$. We prove that $s' \stackrel{a}{\longrightarrow}$. Since $s \preceq_w t$ and $s \stackrel{\tau}{\longrightarrow}^* s'$, there exists a $t'$ such that $t \stackrel{\tau}{\longrightarrow}^* t'$ and $s' \preceq_w t'$. As $(t, u) \models \langle a \rangle \mathtt{tt}$, we infer that $t' \stackrel{a}{\longrightarrow}$. As $a \in \mathcal{U}$ and $s' \preceq_w t'$, it now follows that $s' \stackrel{a}{\longrightarrow}$, which was to be shown.
– CASE $\varphi \equiv \mathbb{\forall}_S \phi$. Assume that $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$. We prove that $((s', u + d), \phi) \in \mathcal{R}$. To this end, note that, as $s \preceq_w t$ and $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$, there is a state $t'$ such that $t \stackrel{\epsilon(d)}{\Longrightarrow}_S t'$ and $s' \preceq_w t'$. As $(t, u)$ satisfies $\mathbb{\forall}_S \phi$, it follows that $(t', u + d)$ satisfies $\phi$. By the definition of $\mathcal{R}$, we conclude that $((s', u + d), \phi) \in \mathcal{R}$, which was to be shown.
– CASE $\varphi \equiv \mathtt{max}(X, \phi)$. Assume that $s \stackrel{\tau}{\longrightarrow}^* s'$. We prove that $((s', u), \phi\{\mathtt{max}(X, \phi)/X\}) \in \mathcal{R}$. To this end, note that, as the relation $\stackrel{\tau}{\longrightarrow}^*$ coincides with $\stackrel{\epsilon(0)}{\Longrightarrow}_\emptyset$, there exists a state $t'$ such that $t \stackrel{\tau}{\longrightarrow}^* t'$ and $s' \preceq_w t'$. Moreover, as $(t, u)$ satisfies $\varphi$, we infer that $(t', u) \models \phi\{\mathtt{max}(X, \phi)/X\}$. By the definition of $\mathcal{R}$, it follows that $((s', u), \phi\{\mathtt{max}(X, \phi)/X\}) \in \mathcal{R}$, which was to be shown.

The claim now follows because $\models$ is the largest satisfiability relation. $\square$

It is now natural to wonder whether the converse of the above result also holds. In particular, we study whether the preorder on states of timed automata induced by the property language $L_{\forall S}$ coincides with $\preceq_w$. We shall now proceed to argue that this is indeed the case if the "more abstract" timed automaton is $\tau$-free and deterministic. By means of a counterexample, we shall also show that the preorder on states induced by the property language is, in general, strictly coarser than $\preceq_w$. In establishing the aforementioned (partial) behavioural characterization result, we shall make an essential use of the notion of *characteristic formula* [SI94].

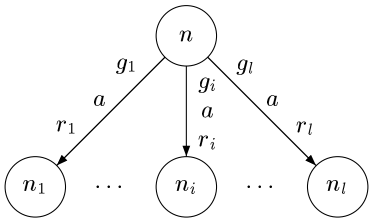



**Figure 9.** Node $n$ of a timed automaton and its $a$-successors.

Consider the portion of a general timed automaton shown in Figure 9. In the figure we can see the nodes that are reachable from node $n$ by an $a$-labelled edge ($g_i$ represents the guard in the edge leading to node $n_i$ and $r_i$ the clocks to be reset in that edge). When it is the case that for every node $n$ and for every action $a$, the guards $g_i$ are disjoint, *i.e.* the condition $g_i \wedge g_j$ is unsatisfiable when $i \neq j$, then the timed automaton $A$ is *deterministic*. (Note that this means that if $a$ is an urgent action, then there is at most one $a$-labelled edge out of node $n$.)

We now proceed to define the characteristic formula for the nodes of a $\tau$-free, deterministic timed automaton with respect to the timed ready simulation preorder introduced above. For the sake of clarity, in the following definition we shall specify recursive formulae using finite systems of recursion equations in lieu of the $\mathtt{max}(X, \varphi)$ construct.

**Definition 6.3 (Characteristic formula for deterministic timed automata).** Let $A$ be a $\tau$-free, deterministic timed automaton. For every node $n$ of $A$, we define the characteristic formula $\phi(n)$

as follows:

$$\phi(n) \stackrel{\mathrm{def}}{=} \forall\!\!\!\forall_{\emptyset}\Big( \bigwedge_{a\in \mathsf{Act}} [a]\Big(\textsc{Enabled}(n,a) \wedge \textsc{Match}(n,a)\Big) \wedge \bigwedge_{a\in \mathcal{U}} \textsc{Out}(n,a)\Big) \tag{5}$$

where

$$\begin{aligned}
\textsc{Enabled}(n,a) &\stackrel{\mathrm{def}}{=} \bigvee_{e\in E(n,a)} g_e \\
\textsc{Match}(n,a) &\stackrel{\mathrm{def}}{=} \bigwedge_{e\in E(n,a)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi(n_e)) \\
\textsc{Out}(n,a) &\stackrel{\mathrm{def}}{=} \textsc{Enabled}(n,a) \Rightarrow \langle a\rangle \mathtt{tt} \ .
\end{aligned}$$

Let $A$ be a $\tau$-free, deterministic timed automaton with initial node $n_0$. We define the characteristic formula of $A$, notation $\phi(A)$, to be $\phi(n_0)$. Note that the set of clocks occurring in $\phi(A)$ is included in the set of clocks of $A$.

Intuitively, the formula $\phi(n)$ requires that, no matter how much a state $s$ delays, and no matter how an action $a$ is performed, then

- there should be at least one $a$-labelled edge of $n$ that is enabled by the current value of the clocks (formula $\textsc{Enabled}(n,a)$). Note that, if $a$ is urgent, then $\textsc{Enabled}(n,a)$ is either $\mathtt{ff}$ (if $n$ has no outgoing $a$-labelled edge) or a tautology;
- the successor state of $s$ satisfies the characteristic formula of the target node of the only $a$-labelled edge of $n$ that is enabled with respect to the current valuation, modulo the appropriate resets of clocks (formula $\textsc{Match}(n,a)$ and determinism of $n$); and
- $s$ has an $a$-labelled transition ($a$ urgent) if $n$ has an $a$-labelled edge (formula $\textsc{Out}(n,a)$).

These intuitive remarks capture the essence of the proof of the following result.

**Theorem 6.4.** *Let $A$ and $B$ be two timed automata. Assume that $B$ is deterministic and $\tau$-free. Then, for every node $n$ of $A$ and $m$ of $B$, and valuations $v, w$ for the clocks in $A$ and $B$, respectively,*

$$\langle n,v\rangle \preceq_w \langle m,w\rangle \ \text{ iff } \ (\langle n,v\rangle, w) \models \phi(m) \ .$$

*Proof.* The proof of this theorem can be found in Appendix C. □

**Corollary 6.5 (Partial behavioural characterization).** *Let $A$ and $B$ be two timed automata, $B$ deterministic and $\tau$-free. Let $\langle n_0,v_0\rangle$ and $\langle m_0,w_0\rangle$ be the initial states of $A$ and $B$ respectively. Then $A \preceq_w B$ iff the set of formulae in $L^-_{\forall S}$ satisfied by $(\langle m_0,w_0\rangle,[K \to 0])$ is included in the set of formulae satisfied by $(\langle n_0,v_0\rangle,[K \to 0])$.*

*Proof.* The "only if" implication follows immediately by Theorem 6.2. To establish the "if" implication we assume that the set of formulae in $L^-_{\forall S}$ satisfied by $(\langle m_0,w_0\rangle, K \to 0)$ is included in the set of formulae satisfied by $(\langle n_0,v_0\rangle, K \to 0)$, and reason as follows. Construct a timed automaton $B'$ which is isomorphic to $B$, and whose collection of clocks is included in $K$. (Note that this is always possible because $K$ is countably infinite.) Let $h$ be an isomorphism from $B$ to $B'$, and let $z_0$ be the initial assignment for the clocks in $B'$. Clearly we have that $\langle m_0,w_0\rangle \preceq_w \langle h(m_0),z_0\rangle \preceq_w \langle m_0,w_0\rangle$. We now argue thus:

$$\begin{aligned}
\langle m_0,w_0\rangle \preceq_w \langle h(m_0),z_0\rangle \quad &\text{iff} \quad (\langle m_0,w_0\rangle,z_0) \models \phi(h(m_0)) \\
&\qquad\qquad \text{(Theorem 6.4)} \\
&\text{implies} \quad (\langle n_0,v_0\rangle,z_0) \models \phi(h(m_0)) \\
&\qquad\qquad \text{(By assumption, since } \phi(h(m_0)) \in L^-_{\forall S} \text{ by Lemma 4.11)} \\
&\text{iff} \quad \langle n_0,v_0\rangle \preceq_w \langle h(m_0),z_0\rangle \\
&\qquad\qquad \text{(Theorem 6.4)} \\
&\text{implies} \quad \langle n_0,v_0\rangle \preceq_w \langle m_0,w_0\rangle \\
&\qquad\qquad (\langle h(m_0),z_0\rangle \preceq_w \langle m_0,w_0\rangle \text{ and transitivity of } \preceq_w) \ .
\end{aligned}$$

This concludes the proof. □

The above result provides a behavioural characterization of the preorder induced by the property language $L_{\forall S}$ over timed automata which holds when the "more abstract" automaton under consideration is deterministic and $\tau$-free. A natural question to ask is whether the property language $L_{\forall S}$ is expressive enough to characterize the preorder $\preceq_w$ over arbitrary timed automata. We shall now show that this is not the case by exhibiting two simple automata without clocks $A$ and $B$ with the following properties:

1. $A \not\preceq_w B$, but
2. for every formula $\varphi \in L_{\forall S}$ and every clock valuation $w$, if $B$ satisfies $\varphi$ with respect to $w$, then so does $A$.

Let $A$ and $B$ be the timed automata associated with the the TCCS processes $a.(b+c+d)$ and $a.(b+c)+a.(b+d)$, respectively, where the actions $a, b, c, d$ are contained in $\mathcal{A}$. It is trivial to see that $a.(b+c+d) \not\preceq_w a.(b+c)+a.(b+d)$. However, we can now argue that:

**Proposition 6.6.** *For every formula $\varphi \in L_{\forall S}$ and every valuation $w$ for the clocks in $K$,*

$$\textit{if } (a.(b+c)+a.(b+d), w) \models \varphi, \textit{ then } (a.(b+c+d), w) \models \varphi.$$

*Proof.* The proof of this proposition can be found in Appendix C. □

Thus the assumption of determinism is necessary in the proof of Corollary 6.5 and of Theorem 6.4.

The reader might also wonder at this point about an apparent mismatch between the definition of the preorder $\preceq_w$ (see Definition 6.1) and that of the characteristic formula construction given in Definition 6.3. In particular, in order for $s \preceq_w t$ to hold, the preorder $\preceq_w$ requires that *every* delay transition of $s$ of the form $s \stackrel{\epsilon(d)}{\Longrightarrow}_S s'$ be matched by some, similarly labelled, delay transition of $t$. On the other hand, the characteristic formula only mentions the $\forall\!\!\!\forall_\emptyset$ quantification over delay transitions. We now proceed to clarify this point by offering an alternative characterization of the preorder $\preceq_w$ for the collection of states of timed automata considered in the statement of Corollary 6.5.

In what follows, we shall use $\preceq_w^\emptyset$ to denote the preorder defined like $\preceq_w$, but which requires clause 2 in Definition 6.1 to hold only for $S = \emptyset$. We now proceed to study the relationships between $\preceq_w$ and $\preceq_w^\emptyset$. In the sequel, we shall say that a state $t$ of a TLTS is *tau-free* is no state reachable from $t$ in zero or more steps affords a $\tau$-labelled transition.

**Proposition 6.7.** *Let $s, t$ be states in a TLTS.*

1. *If $s \preceq_w t$, then $s \preceq_w^\emptyset t$.*
2. *If $s \preceq_w^\emptyset t$ and $t$ is $\tau$-free, then $s \preceq_w t$.*

*Proof.* We only present a proof of the second statement. Assume that $s \preceq_w^\emptyset t$ and $t$ is $\tau$-free. To show that $s \preceq_w t$ holds, it is sufficient to prove that the relation

$$\mathcal{R} \stackrel{\text{def}}{=} \{(s', t') \mid s' \preceq_w^\emptyset t' \text{ and } t' \text{ is } \tau\text{-free}\}$$

satisfies the defining clauses of $\preceq_w$ (see Definition 6.1). To this end, assume that $s' \preceq_w^\emptyset t'$ and $t'$ is $\tau$-free. The only interesting clause in the definition of $\preceq_w$ to check is clause 2. Assume thus that $s' \stackrel{\epsilon(d)}{\Longrightarrow}_S s'_1$ for some $d \in \mathbb{R}_{\geq 0}$ and collection of urgent actions $S$. We shall now show that $t' \stackrel{\epsilon(d)}{\longrightarrow}_S t'_1$ for some $t'_1$ such that $s'_1 \mathrel{\mathcal{R}} t'_1$. The claim is trivial if $d = 0$. Assume thus that $d$ is positive.

First of all, note that, as $s' \stackrel{\epsilon(d)}{\Longrightarrow}_S s'_1$, we have that $s' \stackrel{\epsilon(d)}{\Longrightarrow}_\emptyset s'_1$. Since $s' \preceq_w^\emptyset t'$ and $t'$ is $\tau$-free, there exists a state $t'_1$ such that $t' \stackrel{\epsilon(d)}{\longrightarrow}_\emptyset t'_1$ and $s'_1 \preceq_w^\emptyset t'_1$. We claim that $t' \stackrel{\epsilon(d)}{\longrightarrow}_S t'_1$. Assume, in fact, for the sake of contradiction, that there exists a real number $e \in [0, d[$ and an action $a \in S$ such that $t' \stackrel{\epsilon(e)}{\longrightarrow}_\emptyset \text{delay}(t', e) \stackrel{a}{\longrightarrow}$. Since $a$ is urgent, backward persistence of urgent actions yields that

$t' \xrightarrow{a}$. As $s' \preceq_w^{\emptyset} t'$ and $t'$ is $\tau$-free, it is now easy to see that if $s' \xrightarrow{\tau}^{*} s''$ then $s'' \xrightarrow{a}$. However, as $d > 0$, this contradicts the assumption that $s' \xRightarrow{\epsilon(d)}_S s'_1$.
It follows from the above analysis that, for every real number $e \in [0, d[$ and $a \in S$, $t' \xrightarrow{\epsilon(e)}_{\emptyset} \text{delay}(t', e)$ implies $\text{delay}(t', e) \not\xrightarrow{a}$. Thus $t' \xrightarrow{\epsilon(d)}_S t'_1$ and $s'_1 \preceq_w^{\emptyset} t'_1$, which was to be shown. □

The preorder $\preceq_w^{\emptyset}$ is strictly coarser than $\preceq_w$ over arbitrary timed automata, as shown in the following example.

*Example 6.8.* Consider the timed automata $A$ and $B$ obtained by picking nodes $s_0$ and $t_0$, respectively, as initial nodes in Figure 6. We aim at arguing that

1. $(s_0, [x \to 0]) \not\preceq_w (t_0, [x \to 0])$, but
2. $(s_0, [x \to 0]) \preceq_w^{\emptyset} (t_0, [x \to 0])$.

We examine these two statements in turn.

**Proof of** *1.* We argued in Example 4.9 that $(s_0, [[x \to 0]]) \not\models \forall_{\{a\}}[b]\mathtt{ff}$, but that $(t_0, [[x \to 0]]) \models \forall_{\{a\}}[b]\mathtt{ff}$. In light of Theorem 6.2, it follows that $(s_0, [x \to 0]) \not\preceq_w (t_0, [x \to 0])$.
**Proof of** *2.* Consider the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \{((s_0, [x \to d]), (t_0, [x \to d])) \mid d < 1\} \cup \mathcal{I}$$

where $\mathcal{I}$ denotes the identity relation on states. It is a simple exercise to show that every delay transition indexed by $\emptyset$ from a state of the form $(s_0, [x \to d])$ with $d < 1$ can be matched, up to $\mathcal{R}$, by $(t_0, [x \to d])$. Since $d < 1$, no other transition is enabled from states whose control node is either $s_0$ or $t_0$. Thus $\mathcal{R}$ satisfies the defining clauses of $\preceq_w^{\emptyset}$, and the claim follows.

As a corollary of the results presented in this section, we obtain that timed ready simulation is "testable" in the sense of this paper. In particular, we have shown how the problem of checking the existence of a behavioural relation between states of two timed automata can be recast as a reachability problem that can be efficiently handled by UPPAAL. We envisage that such an approach can, for instance, be applied to yield automatic tool support for the justification of the abstraction steps used in, *e.g.*, [TB97]. In order to take full advantage of this approach, abstraction steps need to be justified using a precongruence relation with respect to the chosen notion of parallel composition. Here we just remark that neither timed simulation nor timed ready simulation is preserved by TCCS parallel composition—which is the one adopted in UPPAAL to combine open systems. However, both the aforementioned relations are preserved by TCCS parallel composition if the more abstract system is $\tau$-free. These are precisely the abstraction steps supported by our method.

## 7 Conclusion

As argued in, *e.g.*, [Wol97], efficient algorithms for deciding reachability questions can be used to tackle many common problems related to verification. In this study, following the lead of [VW86], we have shown how to reduce model-checking of safety and bounded liveness properties expressible in the real-time property language $L_{\forall S}$ to checking for reachability of reject states in suitably constructed test automata. This approach allows us to take full advantage of the core of the computational engine of the tool UPPAAL [BLL+95, LPY97, ABB+01], which consists of a collection of efficient algorithms that can be used to perform reachability analysis over timed automata.

As the property language that we consider is powerful enough to describe characteristic properties [SI94] for nodes of timed automata with respect to (ready) simulation, our approach to model-checking also allows us to reduce the computation of behavioural relations to reachability analysis. Historically, model-checking and reachability analysis were amongst the first problems shown to be decidable for timed automata [ACD93, AD94]. (See also [CY92, AL99].) The decidability of behavioural equivalences and preorders was shown at a later date in [Cer92]. This study may be seen as tracing back the decidability of a behavioural relation, viz. (ready) simulation, to that of the reachability problem via model-checking.

The practical applicability of the approach to model-checking that we have developed in this paper has been tested on a basic CSMA/CD protocol [ABL98]. More experimental activity will be needed to fully test the feasibility of model-checking via reachability testing. So far, all the case studies carried out with the use of UPPAAL (see, *e.g.*, [BGK+96, JLS96, KP95]) seem to support the conclusion that this approach to model-checking can indeed be applied to realistic case studies, but further evidence needs to be accumulated to substantiate this claim. In this process of experimentation, we also expect to further develop a collection of heuristics that can be used to reduce the size of the test automata obtained by means of our automatic translation of formulae into automata.

In this study, we have shown how to translate the formulae in the property language $L_{\forall S}$ into test automata in such a way that model-checking can be reduced to testing for reachability of distinguished reject nodes in the generated automata. Indeed the property language $L_{\forall S}$ is completely expressive with respect to reachability properties, that is we can reduce any reachability property for a composite system $S \parallel T$ to a model-checking problem for $S$.

The results that we have developed show that a timed version of ready simulation is testable, in the sense of this paper. This conclusion seems to be in agreement with the analysis of behavioural relations carried out in [AV93] within the framework of quantales. Whether our results can be justified by means of a general theory à la Abramsky and Vickers is an interesting topic for further theoretical research. It would also be interesting to investigate the connections between our investigations and the seminal study [BIM95], where ready simulation is characterized as the largest precongruence, with respect to all the GSOS definable operations, which is contained in the preorder induced by completed trace inclusion.

**Acknowledgements:** Many thanks to François Laroussinie for his insightful comments on drafts of this paper, and for his thorough proofreading. We thank the anonymous referees for comments that led to improvements in the readability of the paper.

## References


[ABB+01] AMNELL, T., BEHRMANN, G., BENGTSSON, J., D'ARGENIO, P. R., DAVID, A., FEHNKER, A., HUNE, T., JEANNET, B., LARSEN, K. G., MÖLLER, O., PETTERSSON, P., WEISE, C., and YI, W. UPPAAL - *Now, Next, and Future*. In *Proc. Modelling and Verification of Parallel Processes (*MOVEP2k*)*, vol. 2067 of *Lecture Notes in Computer Science Tutorial*, pp. 100–125. Springer-Verlag, 2001.

[ABBL98] ACETO, L., BOUYER, P., BURGUEÑO, A., and LARSEN, K. G. *The Power of Reachability Testing for Timed Automata*. In *Proc. 18th Conf. Found. of Software Technology and Theoretical Computer Science (FST&TCS'98)*, vol. 1530 of *Lecture Notes in Computer Science*, pp. 245–256. Springer-Verlag, 1998. Also available as BRICS Report RS–98–48, Aalborg University, 1998.

[ABL98] ACETO, L., BURGUEÑO, A., and LARSEN, K. G. *Model-Checking via Reachability Testing for Timed Automata*. In *Proc. 4th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, vol. 1384 of *Lecture Notes in Computer Science*, pp. 263–280. Springer-Verlag, 1998. Also available as BRICS Report RS–97–29, Aalborg University, 1997.

[ACD93] ALUR, R., COURCOUBETIS, C., and DILL, D. *Model-Checking in Dense Real-Time*. Information and Computation, vol. 104(1):2–34, 1993.

[AD94] ALUR, R. and DILL, D. *A Theory of Timed Automata*. Theoretical Computer Science, vol. 126(2):183–235, 1994.

[AL99] ACETO, L. and LAROUSSINIE, F. *Is your Model-Checker on Time ?*. In *Proc. 24th Int. Symp. Mathematical Foundations of Computer Science (MFCS'99)*, vol. 1672 of *Lecture Notes in Computer Science*, pp. 125–136. Springer-Verlag, 1999. Full version to appear in the Journal of Logic and Algebraic Programming, 2002.

[And95] ANDERSEN, H. R. *Partial Model-Checking (Extended Abstract)*. In *Proc. 10th IEEE Symp. Logic in Computer Science (LICS'95)*, pp. 398–407. IEEE Computer Society Press, 1995.

[AV93] ABRAMSKY, S. and VICKERS, S. *Quantales, Observational Logic and Process Semantics*. Mathematical Structures in Computer Science, vol. 3(2):161–227, 1993.

[BDFP00] BOUYER, P., DUFOURD, C., FLEURY, E., and PETIT, A. *Are Timed Automata Updatable?*. In *Proc. 12th Int. Conf. Computer Aided Verification (CAV'2000)*, vol. 1855 of *Lecture Notes in Computer Science*, pp. 464–479. Springer-Verlag, 2000.

[BDM+98] BOZGA, M., DAWS, C., MALER, O., OLIVERO, A., TRIPAKIS, S., and YOVINE, S. KRONOS*: a Model-Checking Tool for Real-Time Systems*. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98)*, vol. 1427 of *Lecture Notes in Computer Science*, pp. 546–550. Springer-Verlag, 1998.

[BF99] BÉRARD, B. and FRIBOURG, L. *Automatic Verification of a Parametric Real-Time Program : the ABR Conformance Protocol*. In *Proc. 11th Int. Conf. Computer Aided Verification (CAV'99)*, vol. 1633, pp. 96–107. Springer-Verlag, 1999.

[BGK+96] BENGTSSON, J., GRIFFIOEN, W. D., KRISTOFFERSEN, K. J., LARSEN, K. G., LARSSON, F., PETTERSSON, P., and YI, W. *Verification of an Audio Protocol with Bus Collision Using* UPPAAL. In *Proc. 8th Int. Conf. Computer Aided Verification (CAV'96)*, vol. 1102 of *Lecture Notes in Computer Science*, pp. 244–256. Springer-Verlag, 1996.

[BIM95] BLOOM, B., ISTRAIL, S., and MEYER, A. R. *Bisimulation can't be Traced*. Journal of the Association for Computing Machinery, vol. 42(1):232–268, 1995.

[BLL+95] BENGTSSON, J., LARSEN, K. G., LARSSON, F., PETTERSSON, P., and YI, W. UPPAAL *– a Tool Suite for Automatic Verification of Real-Time Systems*. In *Proc. Workshop Hybrid Systems III: Verification and Control*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 232–243. 1995.

[BLL+98] BENGTSSON, J., LARSEN, K. G., LARSSON, F., PETTERSSON, P., WANG, Y., and WEISE, C. *New Generation of* UPPAAL. In *Proc. Int. Workshop on Software Tools for Technology Transfer*. 1998.

[BS00] BÉRARD, B. and SIERRA, L. *Comparing Verification with* HYTECH*,* KRONOS *and* UPPAAL *on the Railroad Crossing Example*. Research Report LSV–00–2, Lab. Specification and Verification, ENS de Cachan, France, 2000.

[Cer92] CERANS, K. *Decidability of Bisimulation Equivalences for Parallel Timer Processes*. In *Proc. 4th Int. Conf. Computer Aided Verification (CAV'92)*, vol. 663 of *Lecture Notes in Computer Science*, pp. 302–315. Springer-Verlag, 1992.

[CY92] COURCOUBETIS, C. and YANNAKAKIS, M. *Minimum and Maximum Delay Problems in Real-Time Systems*. Formal Methods in System Design, pp. 385–415, 1992.

[DNH84] DE NICOLA, R. and HENNESSY, M. *Testing Equivalences for Processes*. Theoretical Computer Science, vol. 34:83–133, 1984.

[DY95] DAWS, C. and YOVINE, S. *Two Examples of Verification of Multirate Timed Automata with* KRONOS. In *Proc. 16th IEEE Real-Time Systems Symposium (RTSS'95)*, pp. 66–75. IEEE Computer Society Press, 1995.

[GPSS80] GABBAY, D. M., PNUELI, A., SHELAH, S., and STAVI, J. *On the Temporal Analysis of Fairness*. In *Proc. 6th ACM Symp. Principles of Programming Languages*, vol. 1384 of *Lecture Notes in Computer Science*, pp. 163–173. Springer-Verlag, 1980.

[HHWT95] HENZINGER, T. A., HO, P.-H., and WONG-TOI, H. HYTECH*: the Next Generation*. In *Proc. 16th IEEE Real-Time Systems Symposium (RTSS'95)*, pp. 56–65. IEEE Computer Society press, 1995.

[HHWT97] HENZINGER, T., HO, P., and WONG-TOI, H. *HyTech: A Model Checker for Hybrid Systems*. In *Software Tools for Technology Transfer*, pp. 110–122. 1997.

[HK94] HENZINGER, T. A. and KOPKE, P. W. *Undecidability Results for Hybrid Systems*. In *Proc. Workshop on Hybrid Systems and Autonomous Control*. 1994. Also appeared as Cornell University Technical Report TR95-1483.

[HKPV95] HENZINGER, T. A., KOPKE, P. W., PURI, A., and VARAIYA, P. *What's Decidable About Hybrid Automata?*. In *Proc. 27th ACM Symp. Theory of Computing (STOC'95)*, pp. 373–382. 1995. Also appeared as Cornell University Technical Report TR95-1541.

[HWT95] HO, P.-H. and WONG-TOI, H. *Automated Analysis of an Audio Control Protocol*. In *Proc. 7th. Int. Conf. Computer Aided Verification (CAV'95)*, vol. 939 of *Lecture Notes in Computer Science*, pp. 381–394. Springer-Verlag, 1995.

[IKL+00] IVERSEN, T. K., KRISTOFFERSEN, K. J., LARSEN, K. G., LAURSEN, M., MADSEN, R. G., MORTENSEN, S. K., PETTERSSON, P., and THOMASEN, C. B. *Model-Checking Real-Time Control Programs – Verifying LEGO Mindstorms Systems Using* UPPAAL. In *Proc. 12th Euromicro Conf. on Real-Time Systems (ECRTS'00)*. 2000. Also available as BRICS Report RS–99–53, Aalborg University, 1999.

[JLS96] JENSEN, H. E., LARSEN, K. G., and SKOU, A. *Modelling and Analysis of a Collision Avoidance Protocol using SPIN and* UPPAAL. In *Proc. 2nd SPIN Verification Workshop on Algorithms, Applications, Tool Use, Theory*. American Mathematical Society, 1996. Also available as BRICS Report RS–96–24, Aalborg University, 1996.

[Kle56] KLEENE, S. C. *Representation of Events in Nerve Nets and Finite Automata*. In *Automata Studies*, pp. 3–41. Princeton University Press, 1956.

[KP95] KRISTOFFERSEN, K. J. and PETTERSSON, P. *Modelling and Analysis of a Steam Generator Using* UPPAAL. In *Proc. 7th Nordic Workshop on Programming Theory*. 1995.

[Lar90] LARSEN, K. G. *Proof Systems for Satisfiability in Hennessy-Milner Logic with Recursion*. Theoretical Computer Science, vol. 72(2-3):265–288, 1990.

[LL95] LAROUSSINIE, F. and LARSEN, K. G. *Compositional Model-Checking of Real-Time Systems*. In *Proc. 6th Int. Conf. Theory of Concurrency (CONCUR'95)*, vol. 962 of *Lecture Notes in Computer Science*, pp. 529–539. Springer-Verlag, 1995. Also available as BRICS Report RS–95–19, Aalborg University, 1995.

[LL98] LAROUSSINIE, F. and LARSEN, K. G. *CMC: A Tool for Compositional Model-Checking of Real-Time Systems*. In *Proc. IFIP Joint Int. Conf. Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'98)*, pp. 439–456. Kluwer Academic, 1998.

[LLW95] LAROUSSINIE, F., LARSEN, K. G., and WEISE, C. *From Timed Automata to Logic – and Back*. In *Proc. 20th Int. Symp. Mathematical Foundations of Computer Science (MFCS'95)*, vol. 969 of *Lecture Notes in Computer Science*, pp. 27–41. Springer-Verlag, 1995. Also available as BRICS Report RS–95–2, Aalborg University, 1995.

[LPY97] LARSEN, K. G., PETTERSSON, P., and YI, W. UPPAAL *in a Nutshell*. Journal of Software Tools for Technology Transfer, vol. 1(1–2):134–152, 1997.

[LS91] LARSEN, K. G. and SKOU, A. *Bisimulation through Probabilistic Testing*. Information and Computation, vol. 94:1–28, 1991.

[LX91] LARSEN, K. G. and XINXIN, L. *Compositionality through an Operational Semantics of Contexts*. Journal of Logic and Computation, vol. 1:761–795, 1991.

[Mar00] MARGENSTERN, M. *Frontier between Decidability and Undecidability: a Survey*. Theoretical Computer Science, vol. 231(2):217–251, 2000.

[Mil89] MILNER, R. *Communication and Concurrency*. Prentice Hall International, 1989.

[OY93] OLIVERO, A. and YOVINE, S. KRONOS*: a Tool for Verifying Real-Time Systems. User's Guide and Reference Manual*. VERIMAG, Grenoble, France, 1993.

[SI94] STEFFEN, B. and INGÓLFSDÓTTIR, A. *Characteristic Formulae for Processes with Divergence*. Information and Computation, vol. 110(1):149–163, 1994.

[Tar55] TARSKI, A. *A Lattice-Theoretical Fixpoint Theorem and its Applications*. Pacific Journal of Mathematics, vol. 5:285–309, 1955.

[TB97] TAŞIRAN, S. and BRAYTON, R. K. *STARI: A Case Study in Compositional Verification and Hierarchical Timing Verification*. In *Proc. 9th Int. Conf. Computer Aided Verification (CAV'97)*, vol. 1254 of *Lecture Notes in Computer Science*, pp. 191–201. Springer-Verlag, 1997.

[VW86] VARDI, M. Y. and WOLPER, P. *An Automata-Theoretic Approach to Automatic Program Verification*. In *Proc. 1st Annual Symposium on Logic in Computer Science (LICS'86)*, pp. 322–331. IEEE Computer Society Press, 1986.

[Wol97] WOLPER, P. *Where could SPIN go next ? a Unifying Approach to Exploring Infinite State Spaces*. Slides for an invited talk at the *1997 SPIN Workshop*, Enschede, The Netherlands, 1997. Available at the URL `http://www.montefiore.ulg.ac.be/~pw/papers/psfiles/SPIN4-97.ps`.

[Yi90] YI, W. *Real-Time Behaviour of Asynchronous Agents*. In *Proc. 1st Int. Conf. Theory of Concurrency (CONCUR'90)*, vol. 458 of *Lecture Notes in Computer Science*, pp. 502–520. Springer-Verlag, 1990.

[Yi91] YI, W. *A Calculus of Real-Time Systems*. Ph.D. thesis, Chalmers University of Technology, Göteborg, Sweden, 1991.

[Yov97] YOVINE, S. KRONOS*: A Verification Tool for Real-Time Systems*. Journal of Software Tools for Technology Transfer, vol. 1(1–2):123–133, 1997.

## A Proofs omitted from Section 4

**Proposition 4.3.** *Let $s$ be a state in a TLTS $\mathcal{T}$. Then $s$ passes the test in Figure 5 iff for every state $s'$ of $\mathcal{T}$ and delay $d \in \mathbb{R}_{\geq 0}$, $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ implies $s' \stackrel{b}{\not\Longrightarrow}$.*

*Proof.* We prove the two implications separately.

- 'ONLY IF' IMPLICATION: We prove the contrapositive statement. To this end, assume that there exist a state $s'$ and a delay $d \in \mathbb{R}_{\geq 0}$ such that $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ and $s' \stackrel{b}{\Longrightarrow}$. We shall show that $s$ fails the test $T$.
  First of all, note that
  $$(s \| \langle m_0, [k \to 0] \rangle) \backslash \mathsf{Act} \stackrel{\epsilon(d)}{\Longrightarrow} (s' \| \langle m_0, [k \to d] \rangle) \backslash \mathsf{Act}$$
  because $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ and $\langle m_0, [k \to 0] \rangle \stackrel{\epsilon(d)}{\longrightarrow} \langle m_0, [k \to d] \rangle$. Using the assumption that $s' \stackrel{b}{\Longrightarrow} s''$ for some $s''$, we can now continue the above computation thus:
  $$\begin{aligned}(s' \| \langle m_0, [k \to d] \rangle) \backslash \mathsf{Act} &\stackrel{\tau}{\longrightarrow} (s' \| \langle m_2, [k \to 0] \rangle) \backslash \mathsf{Act} \\ &\stackrel{\tau}{\Longrightarrow} (s'' \| \langle m_3, [k \to 0] \rangle) \backslash \mathsf{Act} \\ &\stackrel{\tau}{\longrightarrow} (s'' \| \langle m_T, [k \to 0] \rangle) \backslash \mathsf{Act} \ .\end{aligned}$$
  Hence the reject node of $T$ is reachable, and $s$ fails the test $T$.
- 'IF' IMPLICATION: Again, we prove the contrapositive statement. To this end, assume that $s$ fails the test $T$. We shall show that there exist a state $s'$ and a delay $d \in \mathbb{R}_{\geq 0}$ such that $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s'$ and $s' \stackrel{b}{\Longrightarrow}$.
  Since $s$ fails the test $T$, there is a computation of the form
  $$(s \| \langle m_0, [k \to 0] \rangle) \backslash \mathsf{Act} \longrightarrow^* (s'' \| \langle m_T, [k \to d'] \rangle) \backslash \mathsf{Act} \tag{6}$$
  for some state $s''$ and $d' \in \mathbb{R}_{\geq 0}$. Take one such computation of minimum length. It is not hard to see that the last transition in such a computation must be of the form
  $$(s_2 \| \langle m_3, [k \to 0] \rangle) \backslash \mathsf{Act} \stackrel{\tau}{\longrightarrow} (s_2 \| \langle m_T, [k \to 0] \rangle) \backslash \mathsf{Act}$$
  for some state $s_2$. It follows that $d' = 0$. Hence, (6) has the form
  $$\begin{aligned}(s \| \langle m_0, [k \to 0] \rangle) \backslash \mathsf{Act} &\longrightarrow^* (s_1 \| \langle m_0, [k \to d] \rangle) \backslash \mathsf{Act} \\ &\stackrel{\tau}{\longrightarrow} (s_1 \| \langle m_2, [k \to 0] \rangle) \backslash \mathsf{Act} \\ &\longrightarrow^* (s_2 \| \langle m_3, [k \to 0] \rangle) \backslash \mathsf{Act} \\ &\stackrel{\tau}{\longrightarrow} (s_2 \| \langle m_T, [k \to 0] \rangle) \backslash \mathsf{Act}\end{aligned}$$
  for some $d \in \mathbb{R}_{\geq 0}$ and state $s_1$. The initial fragment of this computation
  $$(s \| \langle m_0, [k \to 0] \rangle) \backslash \mathsf{Act} \longrightarrow^* (s_1 \| \langle m_0, [k \to d] \rangle) \backslash \mathsf{Act}$$
  can only consist of synchronized delays, summing up to $d$, possibly interspersed with $\tau$-labelled transitions from the left-hand component of the parallel composition. Since $a$ is urgent and $\langle m_0, [k \to d''] \rangle \stackrel{\bar{a}}{\longrightarrow}$, for every $d'' \in \mathbb{R}_{\geq 0}$, no $a$ action becomes enabled from $s$ during this delay activity. It follows that $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s_1$. Since the $b$-labelled edge from node $m_2$ does not reset the clock $k$, the computation
  $$(s_1 \| \langle m_2, [k \to 0] \rangle) \backslash \mathsf{Act} \longrightarrow^* (s_2 \| \langle m_3, [k \to 0] \rangle) \backslash \mathsf{Act}$$
  must derive from the synchronization of $s_1 \stackrel{b}{\Longrightarrow} s_2$ and $\langle m_2, [k \to 0] \rangle \stackrel{\bar{b}}{\longrightarrow} \langle m_3, [k \to 0] \rangle$. We can therefore derive that $s \stackrel{\epsilon(d)}{\Longrightarrow}_{\{a\}} s_1 \stackrel{b}{\Longrightarrow} s_2$, which was to be shown.

The proof is now complete. $\square$

**Lemma 4.14.** *Let $t_0$ and $s_0$ be as in Figure 6. Then, for every formula $\varphi \in \mathrm{SBLL}$ and valuation $v$ for the clocks in $K$,*

$$((\langle t_0, [x \to 0]\rangle, v) \models \varphi \textit{ implies } ((\langle s_0, [x \to 0]\rangle, v) \models \varphi \ .$$

*Proof.* We show that the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \{(((\langle s_0, [x \to d]\rangle, v), \varphi) \mid ((\langle t_0, [x \to d]\rangle, v) \models \varphi,\ \varphi \in \mathrm{SBLL} \text{ and } d < 1\} \cup \models$$

is a satisfiability relation. Indeed, the only interesting thing to check is that the defining clauses of $\models$ are met by the pairs $(((\langle s_0, [x \to d]\rangle, v), \varphi)$ when $d < 1$ and $((\langle t_0, [x \to d]\rangle, v) \models \varphi$. This we proceed to do by a case analysis on the form $\varphi$ may take, and only present the details of the proof for the case $\varphi \equiv \forall\!\!\!\forall\phi$. For the other cases, we just remark that:

- the case $\varphi \equiv [c]\phi$ is trivial because, as $d < 1$, there is no outgoing transition from the state $\langle s_0, [x \to d]\rangle$, and
- the case $\varphi \equiv \langle c\rangle\mathtt{tt}$ is vacuous because $\langle t_0, [x \to d]\rangle \not\models \langle c\rangle\mathtt{tt}$,

As promised, we now present the details of the proof for the selected case.

- Case $\varphi \equiv \forall\!\!\!\forall\phi$. Assume that $\langle s_0, [x \to d]\rangle \stackrel{\epsilon(d')}{\Longrightarrow} \langle s, [x \to d'']\rangle$ for some $d', d'' \in \mathbb{R}_{\geq 0}$ and node $s$. We prove that $((\langle s, [x \to d'']\rangle, v + d'), \phi) \in \mathcal{R}$ by distinguishing three cases depending on whether $d + d' < 1$ or $d + d' = 1$ or $d + d' > 1$.
  - Case $d + d' < 1$. In this case, we may infer that $s_0 = s$ and $d'' = d + d'$. Since $\langle t_0, [x \to d]\rangle \stackrel{\epsilon(d')}{\longrightarrow} \langle t_0, [x \to d + d']\rangle$ and $((\langle t_0, [x \to d]\rangle, v) \models \forall\!\!\!\forall\phi$, it follows that

    $$((\langle t_0, [x \to d + d']\rangle, v + d') \models \phi \ .$$

    Using the definition of the relation $\mathcal{R}$, the claim now follows immediately.
  - Case $d + d' = 1$. In this case, we may still infer that $s_0 = s$ and $d'' = d + d'$. Since $d < 1$ and $d + d' = 1$, we obtain that

    $$\langle t_0, [x \to d]\rangle \stackrel{\tau}{\longrightarrow} \langle t_1, [x \to d]\rangle \stackrel{\epsilon(d')}{\longrightarrow} \langle t_1, [x \to d + d']\rangle \stackrel{\tau}{\longrightarrow} \langle s_0, [x \to d + d']\rangle \ .$$

    As $((\langle t_0, [x \to d]\rangle, v) \models \forall\!\!\!\forall\phi$, it follows that $((\langle s_0, [x \to d + d']\rangle, v + d') \models \phi$, and we are done because $\models$ is included in $\mathcal{R}$.
  - Case $d + d' > 1$. In this case, $\langle s, [x \to d'']\rangle$ can take one of two possible forms, *viz.*
    1. $\langle s, [x \to d'']\rangle = \langle s_0, [x \to d + d']\rangle$, or
    2. $\langle s, [x \to d'']\rangle = \langle s_1, [x \to d + d']\rangle$.

    If 1 holds then, reasoning as above, we may show that $(((\langle s_0, [x \to d + d']\rangle, v + d'), \phi)$ is contained in $\mathcal{R}$. If 2 holds then, as $d < 1$ and $d + d' > 1$, we may infer that

    $$\langle t_0, [x \to d]\rangle \stackrel{\epsilon(d')}{\Longrightarrow} \langle s_1, [x \to d + d']\rangle \ .$$

    Since $((\langle t_0, [x \to d]\rangle, v) \models \forall\!\!\!\forall\phi$, it follows that $((\langle s_1, [x \to d + d']\rangle, v + d')$ satisfies $\phi$. The inclusion of $\models$ in $\mathcal{R}$ now yields the claim.

The remaining cases of the proof are similar, and are left to the reader. $\square$

**Proposition 4.17.**

1. *The formula $\langle a\rangle\mathtt{tt}$ is not testable over states of TLTSs, regardless of whether $a$ is urgent or not.*
2. *The formula $\langle a\rangle\mathtt{tt}$ ($a$ not urgent) is not testable over states of timed automata.*
3. *The formula $[a]\mathtt{ff} \vee [b]\mathtt{ff}$ ($a, b$ not urgent) is not testable over states of timed automata.*

*Proof.* We prove the three statements separately.

1. Assume, towards a contradiction, that a test automaton $T = \langle \mathsf{Act}_\tau, N, N_T, m_0, C, C_0, E\rangle$ tests for the formula $\langle a\rangle\mathtt{tt}$. Consider the TLTS consisting of the single state $s$ and of the transition $s \xrightarrow{\epsilon(0)} s$. Let $u$ be an arbitrary valuation for the formula clocks. Since $(s,u)$ does not satisfy $\langle a\rangle\mathtt{tt}$, the extended state $(s,u)$ must fail the test $T$ — that is, there is a reject node of $T$ which is reachable from the state $(s \parallel \langle m_0, [C_0 \to 0](u \upharpoonright C)\rangle)\backslash\mathsf{Act}$, where $m_0$ is the initial node of $T$. As $s$ only affords a $0$-delay transition, the reader will easily realize that this means that there is a sequence of $\tau$-labelled transitions leading from $\langle m_0, [C_0 \to 0](u \upharpoonright C)\rangle$ to a reject node of $T$. Consider now the TLTS whose only state is $t$, and whose transitions are

$$t \xrightarrow{\epsilon(0)} t \text{ and } t \xrightarrow{a} t \ .$$

   Obviously $(t,u) \models \langle a\rangle\mathtt{tt}$ holds. However, $(t,u)$ fails the test $T$ because $T$ can independently enter a reject node by performing a sequence of $\tau$-labelled transitions. This contradicts our assumption that $T$ tests for the formula $\langle a\rangle\mathtt{tt}$.
2. (SKETCH.) Assume, towards a contradiction, that a test automaton $T$ tests for the formula $\langle a\rangle\mathtt{tt}$ ($a$ not urgent) over states of timed automata. Then the timed automaton depicted in Figure 8(a) must fail the $T$-test. Using the assumption that $a$ is *not* urgent, we can now show by analyzing an arbitrary computation leading to the reject node of $T$ in $(n_0 \parallel T)\backslash\mathsf{Act}$ that a reject node in $T$ can also be reached in $(m_0 \parallel T)\backslash\mathsf{Act}$, where $m_0$ is the initial node of the timed automaton depicted in Figure 8(b). As $m_0$ obviously satisfies the formula $\langle a\rangle\mathtt{tt}$, this contradicts the assumption that $T$ tests for $\langle a\rangle\mathtt{tt}$.
3. (SKETCH.) Assume, towards a contradiction, that a test automaton $T$ tests for the formula $[a]\mathtt{ff} \vee [b]\mathtt{ff}$. Then the timed automaton associated with the TCCS agent $a + b$ [Yi90] must fail the $T$-test. Using the assumption that $T$ tests for $[a]\mathtt{ff} \vee [b]\mathtt{ff}$, by a careful analysis of an arbitrary computation leading to the reject node of $T$ in $((a+b) \parallel T)\backslash\{a,b\}$, we infer that either (A) such a computation only contains delays and $\tau$-labelled transitions from the tester $T$, or (B) that it must involve one $a$ or $b$-synchronization, possibly preceded and followed by several $\tau$ or delay transitions from the tester $T$.
   If (A) applies, then the timed automaton associated with the TCCS process NIL [Yi90] will also fail the test $T$. Since such a timed automaton obviously satisfies the property $[a]\mathtt{ff} \vee [b]\mathtt{ff}$, this contradicts our assumption that $T$ tests for $[a]\mathtt{ff} \vee [b]\mathtt{ff}$.
   If (B) applies, then either the timed automaton associated with the TCCS process $a$ or the one associated with the process $b$ will also fail the test $T$. Again, since both such timed automata obviously satisfy the property $[a]\mathtt{ff} \vee [b]\mathtt{ff}$, this contradicts our assumption that $T$ tests for $[a]\mathtt{ff} \vee [b]\mathtt{ff}$. ☐

## B Compositionality of $L_{\forall S}$

**Theorem 5.6.** *Let $\varphi$ be a closed formula in $L^-_{\forall S}$. Suppose that $s$ is a state of a TLTS, and $n'$ is a state of a timed automaton over set of clocks $C$. Suppose furthermore that $u$ and $v'$ are valuations for the disjoint set of clocks* $\mathtt{clocks}(\varphi)$ *and $C$, respectively. Then*

$$(s \parallel \langle n', v'\rangle, u) \models \varphi \iff (s, v' : u) \models \varphi/n' \ .$$

*Proof.* We prove the two implications separately.

– IF IMPLICATION. We aim at showing that $(s, v' : u) \models \varphi/n'$ implies $(s \parallel \langle n', v'\rangle, u) \models \varphi$. To this end, it is sufficient to prove that the following relation $\mathcal{R}$ satisfies the defining clauses of $\models$:

$$\mathcal{R} \stackrel{\text{def}}{=} \{((s \parallel \langle n', v'\rangle, u), \varphi) \mid (s, v' : u) \models \varphi/n'\}$$

  Assume that $((s \parallel \langle n', v'\rangle, u), \varphi) \in \mathcal{R}$. We prove that $\mathcal{R}$ is a satisfiability relation by a case analysis on the form $\varphi$ may take.

- CASE $\varphi \equiv \mathtt{ff}$. This case is vacuous.
- CASE $\varphi \equiv \phi_1 \wedge \phi_2$. Assume that

$$s \parallel \langle n', v' \rangle \xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle \tag{7}$$

Let $k$ be the number of steps in which the second component participates during computation (7). By induction on $k$, we prove that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi_j) \in \mathcal{R}$ for $j = 1, 2$.

  * BASE CASE. If $k = 0$ then $n' = n'_1$, $v' = v'_1$ and $s \xrightarrow{\tau}^* s_1$. Since $(s, v' : u) \models \varphi/n'$, by the definition of the quotient formula $\varphi/n'$ we obtain that $(s_1, v' : u) \models \phi_1/n'$ and $(s_1, v' : u) \models \phi_2/n'$. By the definition of $\mathcal{R}$, it follows that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi_j) \in \mathcal{R}$ for $j = 1, 2$, which was to be shown.
  * INDUCTIVE STEP. Suppose that the result is proved for $k_0$ and that $k = k_0 + 1$. Computation (7) has the following form:

$$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}^* s_3 \parallel \langle n', v' \rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle \\ &\xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

  We proceed with the proof by analyzing the two possible forms transition

$$s_3 \parallel \langle n', v' \rangle \xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle$$

  may take.

    · CASE: $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$ and $s_3 = s_2$. In this case, there must be an edge $e \in E(n', \tau)$ such that $n'_2 = n'_e$, $v'_2 = [r_e \to 0]v'$ and $g_e(v')$ is true. Since $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}^* s_3$, by the definition of the quotient formula $\varphi/n'$ we may infer that $(s_3, [r_e \to 0]v' : u) \models \varphi/n'_e$. By the definition of the relation $\mathcal{R}$, it follows that

$$(s_2 \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$$

    We may now apply the inductive hypothesis to the computation

$$s_2 \parallel \langle n'_2, v'_2 \rangle \xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle$$

    to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi_j) \in \mathcal{R}$ for $j = 1, 2$, which was to be shown.

    · CASE: $\langle n', v' \rangle \xrightarrow{b} \langle n'_2, v'_2 \rangle$ and $s_3 \xrightarrow{\bar{b}} s_2$ for some $b \in \mathsf{Act}$. The proof follows the lines of the one for the previous case, and is left to the reader.

  This completes the inductive argument and the proof for the case $\varphi \equiv \phi_1 \wedge \phi_2$.
- CASE $\varphi \equiv g \vee \phi$. Assume that

$$s \parallel \langle n', v' \rangle \xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle \tag{8}$$

Let $k$ be the number of steps in which the second component participates during computation (8). By induction on $k$, we prove that either $g(v'_1 : u)$ holds or $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$.

  * BASE CASE. If $k = 0$ then $n' = n'_1$, $v' = v'_1$ and $s \xrightarrow{\tau}^* s_1$. Since $(s, v' : u) \models \varphi/n'$, by definition of the quotient formula $\varphi/n'$ we obtain that either $g(v'_1 : u)$ holds or $(s_1, v' : u) \models \phi/n'$. By the definition of $\mathcal{R}$, it follows that either $g(v'_1 : u)$ holds or $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$, which was to be shown.
  * INDUCTIVE STEP. Suppose that the result is proved for $k_0$ and that $k = k_0 + 1$. Computation (8) has the following form:

$$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}^* s_3 \parallel \langle n', v' \rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle \\ &\xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

We proceed with the proof by analyzing the two possible forms transition

$$s_3 \parallel \langle n', v' \rangle \xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle$$

may take.

- CASE: $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$ and $s_3 = s_2$. In this case, there must be an edge $e \in E(n', \tau)$ such that $n'_2 = n'_e$, $v'_2 = [r_e \to 0]v'$ and $g_e(v')$ is true. Since $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}^* s_3$, by the definition of the quotient formula $\varphi/n'$ we may infer that $(s_3, [r_e \to 0]v' : u) \models \varphi/n'_e$. By the definition of the relation $\mathcal{R}$, it follows that

  $$(s_2 \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$$

  We may now apply the inductive hypothesis to the computation

  $$s_2 \parallel \langle n'_2, v'_2 \rangle \xrightarrow{\tau}^* s_1 \parallel \langle n'_1, v'_1 \rangle$$

  to infer that either $g(v'_1 : u)$ holds or $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$, which was to be shown.
- CASE: $\langle n', v' \rangle \xrightarrow{b} \langle n'_2, v'_2 \rangle)$ and $s_3 \xrightarrow{\bar{b}} s_2$ for some $b \in \mathsf{Act}$. The proof follows the lines of the one for the previous case, and is left to the reader.

This completes the inductive argument and the proof for the case $\varphi \equiv q \vee \phi$.

- CASE $\varphi \equiv [a]\phi$. Assume that

$$s \parallel \langle n', v' \rangle \xRightarrow{a} s_1 \parallel \langle n'_1, v'_1 \rangle \tag{9}$$

We prove that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$ by induction on the number of steps $k$ in which the second component participates during computation (9).

* BASE CASE. If $k = 0$ then $n' = n'_1$, $v' = v'_1$ and $s \xRightarrow{a} s_1$. Since $(s, v' : u) \models \varphi/n'$, by definition of the quotient formula $\varphi/n'$ we obtain that $(s_1, v' : u) \models \phi/n'$. By the definition of $\mathcal{R}$, it follows that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$, which was to be shown.
* INDUCTIVE STEP. Suppose that the result is proved for $k_0$ and that $k = k_0 + 1$. We proceed with the proof by considering the three possible forms computation (9) may take.
  - Assume that computation (9) takes the form

    $$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}^* s_1 \parallel \langle n', v' \rangle \\ &\xrightarrow{a} s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

    because $s \xrightarrow{\tau}^* s_1$ and $\langle n', v' \rangle \xrightarrow{a} \langle n'_1, v'_1 \rangle$. Since $\langle n', v' \rangle \xrightarrow{a} \langle n'_1, v'_1 \rangle$, there is an edge $e \in E(n', a)$ such that $g_e(v')$ holds, $n'_1 = n'_e$ and $v'_1 = [r_e \to 0]v'$. As $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}^* s_1$, the definition of the quotient formula $\varphi/n'$ now yields that $(s_1, v'_1 : u) \models \phi/n'_1$. By the definition of $\mathcal{R}$, it follows that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$, which was to be shown.
  - Assume that computation (9) takes the form

    $$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}^* s_2 \parallel \langle n', v' \rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle \\ &\xRightarrow{a} s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

    because $s \xrightarrow{\tau}^* s_2$ and $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$. Since $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$, there is an edge $e \in E(n', \tau)$ such that $g_e(v')$ holds, $n'_2 = n'_e$ and $v'_2 = [r_e \to 0]v'$. As $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}^* s_2$, the definition of the quotient formula $\varphi/n'$ now yields that $(s_2, v'_2 : u) \models \varphi/n'_2$. By the definition of $\mathcal{R}$, it follows that $((s_2 \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$. We may therefore apply the inductive hypothesis to the computation $s_2 \parallel \langle n'_2, v'_2 \rangle \xRightarrow{a} s_1 \parallel \langle n'_1, v'_1 \rangle$ to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi) \in \mathcal{R}$, which was to be shown.

· Assume that computation (9) takes the form

$$\begin{aligned} s \parallel \langle n', v'\rangle &\xrightarrow{\tau}^* s_3 \parallel \langle n', v'\rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n_2', v_2'\rangle \\ &\xRightarrow{a} s_1 \parallel \langle n_1', v_1'\rangle \end{aligned}$$

because $\langle n', v'\rangle \xrightarrow{b} \langle n_2', v_2'\rangle$ and $s \xrightarrow{\tau}^* s_3 \xrightarrow{\bar{b}} s_2$ for some action $b$ in Act. (Note that $b$ can be $a$ itself.) Since $\langle n', v'\rangle \xrightarrow{b} \langle n_2', v_2'\rangle$, there is an edge $e \in E(n', b)$ such that $g_e(v')$ holds, $n_2' = n_e'$ and $v_2' = [r_e \to 0]v'$. As $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}^* s_3$, the definition of the quotient formula $\varphi/n'$ now yields that $(s_2, v_2' : u) \models \varphi/n_2'$. By the definition of $\mathcal{R}$, it follows that $((s_2 \parallel \langle n_2', v_2'\rangle, u), \varphi) \in \mathcal{R}$. We may therefore apply the inductive hypothesis to the computation $s_2 \parallel \langle n_2', v_2'\rangle \xRightarrow{a} s_1 \parallel \langle n_1', v_1'\rangle$ to infer that $((s_1 \parallel \langle n_1', v_1'\rangle, u), \phi) \in \mathcal{R}$, which was to be shown.

This completes the inductive argument and the proof for the case $\varphi \equiv [a]\phi$.

- CASE $\varphi = x \ \underline{\mathtt{in}}\ \phi$. Assume that $(s, u) \parallel ((\langle n', v'\rangle, u) \xrightarrow{\tau}^* (s_1, u) \parallel ((\langle n_1', v_1'\rangle, u)$. We prove that $((s_1, u) \parallel ((\langle n_1', v_1'\rangle, [x \leftarrow 0]u), \phi) \in \mathcal{R}$. Let $k$ be the number of steps of the second component during the computation. The proof will be done by induction on $k$. If $k = 0$, then the second component stays in the location $n'$, and $(s, v' : u) \xrightarrow{\tau}^* (s_1, v_1' : u)$. By definition of the quotient, $(s_1, v_1' : [x \leftarrow 0]u) \models \phi/n'$, and thus

$$((s_1, [x \leftarrow 0]u) \parallel ((\langle n', v_1'\rangle, [x \leftarrow 0]u), \phi) \in \mathcal{R}$$

Now, assume that $k = k_0 + 1$ and that we are done for $k_0$. Then the computation has the following form:

$$\begin{aligned} (s, u) \parallel ((\langle n', v'\rangle, u) &\xrightarrow{\tau}^* (s_3, u) \parallel ((\langle n', v'\rangle, u) \\ &\xrightarrow{\tau} (s_2, u) \parallel ((\langle n_2', v_2'\rangle, u) \\ &\xrightarrow{\tau}^* (s_1, u) \parallel ((\langle n_1', v_1'\rangle, u) \end{aligned}$$

As in the previous cases, there are two cases for the first step of the second component, i.e. for $(s_3, u) \parallel ((\langle n', v'\rangle, u) \xrightarrow{\tau} (s_2, u) \parallel ((\langle n_2', v_2'\rangle, u)$ : the second component does a $\tau$-action or he does a $b$ and synchronizes on $b$ with the first component. So, the proof is similar to the others, and the result follows by induction hypothesis.
- CASE $\varphi = \forall\!\!\!\forall_S \phi$. Assume that

$$s \parallel \langle n', v'\rangle \xRightarrow{\epsilon(d)}_S s_1 \parallel \langle n_1', v_1'\rangle \tag{10}$$

for some $d \geq 0$. We consider the cases $S \cap \mathcal{U}(n') = \emptyset$ and $S \cap \mathcal{U}(n') \neq \emptyset$ separately.
  * CASE: $S \cap \mathcal{U}(n') = \emptyset$. We prove that $((s_1 \parallel \langle n_1', v_1'\rangle, u + d), \phi) \in \mathcal{R}$ by induction on the number of steps $k$ in which the second component participates during computation (10).
    · BASE CASE. If $k = 0$ then it must be the case that $d = 0$, $n' = n_1'$, $v' = v_1'$ and $s \xRightarrow{\epsilon(0)}_S s_1$. As the transition relation $\xRightarrow{\epsilon(0)}_S$ coincides with $\xrightarrow{\tau}^*$ for every index set $S$, it follows that $s \xRightarrow{\epsilon(0)}_{S \cup \overline{\mathcal{U}(n')}} s_1$. Since $(s, v' : u) \models \varphi/n'$, by definition of the quotient formula $\varphi/n'$ we obtain that $(s_1, v' : u) \models \phi/n'$. By the definition of $\mathcal{R}$, it follows that $((s_1 \parallel \langle n_1', v_1'\rangle, u), \phi) \in \mathcal{R}$, which was to be shown.
    · INDUCTIVE STEP. Suppose that the result is proved for $k_0$ and that $k = k_0 + 1$. We proceed with the proof by considering the three possible forms computation (10) may take. 1. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v'\rangle &\xrightarrow{\tau}^* s_2 \parallel \langle n', v'\rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n_2', v_2'\rangle \\ &\xRightarrow{\epsilon(d)}_S s_1 \parallel \langle n_1', v_1'\rangle \end{aligned}$$

because $s \xrightarrow{\tau}{}^{*} s_2$ and $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$. Since $\langle n', v' \rangle \xrightarrow{\tau} \langle n'_2, v'_2 \rangle$, there is an edge $e \in E(n', \tau)$ such that $g_e(v')$ holds, $n'_2 = n'_e$ and $v'_2 = [r_e \to 0]v'$. As $(s, v' : u) \models \varphi/n'$ and $s \xrightarrow{\tau}{}^{*} s_2$, the definition of the quotient formula $\varphi/n'$ now yields that $(s_2, v'_2 : u) \models \varphi/n'_2$. By the definition of $\mathcal{R}$, it follows that

$$((s_2 \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$$

We may therefore apply the inductive hypothesis to the computation

$$s_2 \parallel \langle n'_2, v'_2 \rangle \overset{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle$$

to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u + d), \phi) \in \mathcal{R}$, which was to be shown.

2. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}{}^{*} s_3 \parallel \langle n', v' \rangle \\ &\xrightarrow{\epsilon(d')}_S s_2 \parallel \langle n', v' + d' \rangle \\ &\overset{\epsilon(d'')}{\Longrightarrow} s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $s \xrightarrow{\tau}{}^{*} s_3 \xrightarrow{\epsilon(d')}_S s_2$, $\langle n', v' \rangle \xrightarrow{\epsilon(d')}_S \langle n', v' + d' \rangle$ and $d = d' + d''$. Since $s_3 \parallel \langle n', v' \rangle \xrightarrow{\epsilon(d')}_S s_2 \parallel \langle n', v' + d' \rangle$, we also have that either $d' = 0$ or $s_3 \stackrel{\bar{a}}{\not\longrightarrow}$, for every $a \in \mathcal{U}(n')$. In both these cases, we may infer that $s \overset{\epsilon(d')}{\Longrightarrow}_{S \cup \overline{\mathcal{U}(n')}} s_2$. Since the formula $\varphi/n'$ is equivalent to $\forall\!\!\!\forall_{S \cup \overline{\mathcal{U}(n')}}(\varphi/n')$ over states of TLTSs (Lemma 4.8(2)) and $(s, v' : u) \models \varphi/n'$, we obtain that $(s_2, v' + d' : u + d') \models \varphi/n'$. By the definition of $\mathcal{R}$, it follows that $((s_2 \parallel \langle n', v' + d' \rangle, u + d'), \varphi) \in \mathcal{R}$. We may therefore apply the inductive hypothesis to the computation $s_2 \parallel \langle n', v' + d' \rangle \overset{\epsilon(d'')}{\Longrightarrow} s_1 \parallel \langle n'_1, v'_1 \rangle$ to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u + d' + d''), \phi) \in \mathcal{R}$, which was to be shown.

3. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau}{}^{*} s_3 \parallel \langle n', v' \rangle \\ &\xrightarrow{\tau} s_2 \parallel \langle n'_2, v'_2 \rangle \\ &\overset{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $\langle n', v' \rangle \xrightarrow{b} \langle n'_2, v'_2 \rangle$ and $s \xrightarrow{\tau}{}^{*} s_3 \xrightarrow{\bar{b}} s_2$ for some action $b$ in Act. The proof for this sub-case follows the lines of the one for sub-case 1 above, and is left to the reader.

This completes the inductive argument and the proof for the case $S \cap \mathcal{U}(n') = \emptyset$.

∗ CASE: $S \cap \mathcal{U}(n') \neq \emptyset$. We proceed by induction on the number of steps in computation (10).

· BASE CASE. Assume that $s \parallel \langle n', v' \rangle \xrightarrow{\epsilon(d)}_S s_1 \parallel \langle n'_1, v'_1 \rangle$. This implies, among other things, that $s \xrightarrow{\epsilon(d)}_S s_1$, $\langle n', v' \rangle \xrightarrow{\epsilon(d)}_S \langle n'_1, v'_1 \rangle$ and $\langle n'_1, v'_1 \rangle = \langle n', v' + d \rangle$. As $S \cap \mathcal{U}(n') \neq \emptyset$, state $\langle n', v' \rangle$ can perform an urgent action in the set $S$. This yields that $d = 0$, and, therefore, that $v'_1 = v'$ and $s = s_1$. Since $(s, v' : u) \models \varphi/n'$, by the definition of the quotient formula we obtain that $(s, v' : u) \models \phi/n'$. The definition of $\mathcal{R}$ now yields

$$((s \parallel \langle n', v' \rangle, u), \phi) \in \mathcal{R}$$

which was to be shown.

· INDUCTIVE STEP. We proceed with the proof by considering the four possible forms computation (10) may take. 1. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle &\xrightarrow{\tau} s_2 \parallel \langle n', v' \rangle \\ &\overset{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $s \stackrel{\tau}{\longrightarrow} s_2$. As $(s, v' : u) \models \varphi/n'$ and $s \stackrel{\tau}{\longrightarrow} s_2$, Proposition 4.6 yields that $(s_2, v'_2 : u) \models \varphi/n'_2$. By the definition of $\mathcal{R}$, it follows that

$$((s_2 \parallel \langle n', v' \rangle, u), \varphi) \in \mathcal{R}$$

We may therefore apply the inductive hypothesis to the computation

$$s_2 \parallel \langle n', v' \rangle \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle$$

to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u + d), \phi) \in \mathcal{R}$, which was to be shown.
2. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle & \stackrel{\tau}{\longrightarrow} s_1 \parallel \langle n'_2, v'_2 \rangle \\ & \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $\langle n', v' \rangle \stackrel{\tau}{\longrightarrow} \langle n'_2, v'_2 \rangle$. Since $\langle n', v' \rangle \stackrel{\tau}{\longrightarrow} \langle n'_2, v'_2 \rangle$, there is an edge $e \in E(n', \tau)$ such that $g_e(v')$ holds, $n'_2 = n'_e$ and $v'_2 = [r_e \rightarrow 0]v'$. As $(s, v' : u) \models \varphi/n'$, the definition of the quotient formula $\varphi/n'$ now yields that $(s, v'_2 : u) \models \varphi/n'_2$. By the definition of $\mathcal{R}$, it follows that

$$((s \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$$

We may therefore apply the inductive hypothesis to the computation

$$s \parallel \langle n'_2, v'_2 \rangle \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle$$

to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u + d), \phi) \in \mathcal{R}$, which was to be shown.
3. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle & \stackrel{\epsilon(d')}{\longrightarrow}_S s_2 \parallel \langle n', v' + d' \rangle \\ & \stackrel{\epsilon(d'')}{\Longrightarrow} s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $s \stackrel{\epsilon(d')}{\longrightarrow}_S s_2$, $\langle n', v' \rangle \stackrel{\epsilon(d')}{\longrightarrow}_S \langle n', v' + d' \rangle$ and $d = d' + d''$. As $S \cap \mathcal{U}(n') \neq \emptyset$, state $\langle n', v' \rangle$ can perform an urgent action in the set $S$. This yields that $d' = 0$, and, therefore, that $s = s_2$ and $d'' = d$. We may therefore apply the inductive hypothesis to the computation $s \parallel \langle n', v' \rangle \stackrel{\epsilon(d)}{\Longrightarrow} s_1 \parallel \langle n'_1, v'_1 \rangle$ to infer that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u + d' + d''), \phi) \in \mathcal{R}$, which was to be shown.
4. Assume that computation (10) takes the form

$$\begin{aligned} s \parallel \langle n', v' \rangle & \stackrel{\tau}{\longrightarrow} s_2 \parallel \langle n'_2, v'_2 \rangle \\ & \stackrel{\epsilon(d)}{\Longrightarrow}_S s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

because $\langle n', v' \rangle \stackrel{b}{\longrightarrow} \langle n'_2, v'_2 \rangle$ and $s \stackrel{\bar{b}}{\longrightarrow} s_2$ for some action $b$ in Act. The proof for this sub-case follows the lines of the one for case 1, and is left to the reader.

This completes the inductive argument and the proof for the case $S \cap \mathcal{U}(n') = \emptyset$.

The proof for the case $\varphi \equiv \forall\!\!\!\forall_S \phi$ is now complete.

- CASE $\varphi \equiv \mathtt{max}(X, \phi)$. Assume that

$$s \parallel \langle n', v' \rangle \stackrel{\tau}{\longrightarrow}^* s_1 \parallel \langle n'_1, v'_1 \rangle \tag{11}$$

We prove that $((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi\{\mathtt{max}(X, \phi)/X\}) \in \mathcal{R}$. This we show by induction on the number $k$ of steps of computation (11) in which the second component participates.

* BASE CASE. If $k = 0$, then it must be the case that $s \stackrel{\tau}{\longrightarrow}^* s_1$, $n'_1 = n'$ and $v'_1 = v'$. Since $(s, v' : u) \models \varphi/n'$, we may infer that

$$(s_1, v' : u) \models (\phi\{\mathtt{max}(X, \phi)/X\})/n'$$

The definition of $\mathcal{R}$ now yields

$$((s_1 \parallel \langle n', v' \rangle, u), \phi\{\mathtt{max}(X, \phi)/X\}) \in \mathcal{R}$$

which was to be shown.

* INDUCTIVE STEP. Suppose that the result is proved for $k_0$ and that $k = k_0 + 1$. Computation (11) has the following form:

$$\begin{aligned} s \parallel \langle n', v' \rangle &\stackrel{\tau}{\longrightarrow}^* s_3 \parallel \langle n', v' \rangle \\ &\stackrel{\tau}{\longrightarrow} s_2 \parallel \langle n'_2, v'_2 \rangle \\ &\stackrel{\tau}{\longrightarrow}^* s_1 \parallel \langle n'_1, v'_1 \rangle \end{aligned}$$

We proceed with the proof by analyzing the two possible forms transition

$$s_3 \parallel \langle n', v' \rangle \stackrel{\tau}{\longrightarrow} s_2 \parallel \langle n'_2, v'_2 \rangle$$

may take.

  · CASE: $\langle n', v' \rangle \stackrel{\tau}{\longrightarrow} \langle n'_2, v'_2 \rangle$ and $s_3 = s_2$. In this case, there must be an edge $e \in E(n', \tau)$ such that $n'_2 = n'_e$, $v'_2 = [r_e \to 0]v'$ and $g_e(v')$ is true. Since $(s, v' : u) \models \varphi/n'$ and $s \stackrel{\tau}{\longrightarrow}^* s_3$, by the definition of the quotient formula $\varphi/n'$ we may infer that $(s_3, [r_e \to 0]v' : u) \models \varphi/n'_e$. By the definition of the relation $\mathcal{R}$, it follows that

$$(s_2 \parallel \langle n'_2, v'_2 \rangle, u), \varphi) \in \mathcal{R}$$

  We may now apply the inductive hypothesis to the computation

$$s_2 \parallel \langle n'_2, v'_2 \rangle \stackrel{\tau}{\longrightarrow}^* s_1 \parallel \langle n'_1, v'_1 \rangle$$

  to infer that

$$((s_1 \parallel \langle n'_1, v'_1 \rangle, u), \phi\{\mathtt{max}(X, \phi)/X\}) \in \mathcal{R}$$

  which was to be shown.

  · CASE: $\langle n', v' \rangle \stackrel{b}{\longrightarrow} \langle n'_2, v'_2) \rangle$ and $s_3 \stackrel{\bar{b}}{\longrightarrow} s_2$ for some $b \in \mathrm{Act}$. The proof follows the lines of the one for the previous case, and is left to the reader.

This completes the inductive argument and the proof for the case $\varphi \equiv \mathtt{max}(X, \phi)$.

This completes the proof of the 'if implication'.

– ONLY IF IMPLICATION: We aim at showing that $(s \parallel \langle n', v' \rangle, u') \models \varphi$ implies $(s, v' : u) \models \varphi/n'$. We just need to prove that the environment $\rho$ defined as follows for every formula $\varphi$ and every node $n'$

$$[\![\varphi/n']\!]\rho \stackrel{\text{def}}{=} \{(s, v' : u) \mid (s \parallel \langle n', v' \rangle, u) \models \varphi\}$$

is a post-fixed point of the monotonic functional on environments induced by the definition of the quotient (see Table 5) [Lar90, SI94].
Assume that $(s, v' : u) \in [\![\varphi/n']\!]\rho$. We wish to prove that the state $(s, v' : u)$ is contained in the interpretation of the right-hand side of the formula defining $\varphi/n'$ (see Table 5). We proceed by a case analysis on the form that $\varphi$ may take.

- CASE $\varphi \equiv \mathtt{ff}$. This case is vacuous.

- CASE $\varphi \equiv \phi_1 \wedge \phi_2$. We prove that

$$(s, v' : u) \in \left[\!\left[ \begin{array}{c} (\phi_1/n') \wedge (\phi_2/n') \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e)) \\ \wedge \\ \bigwedge_{b \in \mathsf{Act}} \bigwedge_{e \in E(n',b)} (g_e \ \underline{\mathtt{in}}\ [\bar{b}]\, (r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))) \end{array} \right]\!\right] \rho$$

where $[\![\psi]\!]\rho$ is the set of states satisfying $\psi$ in the environment $\rho$. To this end, assume that $s \xrightarrow{\tau}^* s_1$. We aim at proving that $(s_1, v' : u) \in [\![\psi]\!]\rho$, for every formula $\psi$ in the set

$$\{\phi_1/n',\, \phi_2/n'\} \cup \bigcup_{e \in E(n',\tau)} \{g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e)\} \cup \bigcup_{b \in \mathsf{Act}} \bigcup_{e \in E(n',b)} \{g_e \Rightarrow [\bar{b}]\, (r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))\}$$

For example, let us consider the formula $\psi \equiv g_e \Rightarrow [\bar{b}]\, (r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))$, for some action $b$ and $e \in E(n', b)$. Assume that $g_e(v')$ is true and that $s_1 \overset{\bar{b}}{\Longrightarrow}^* s_2$. Then $s$ and $\langle n', v'\rangle$ can synchronize on $b$ yielding the following computation from $s \| \langle n', v'\rangle$:

$$s \parallel \langle n', v'\rangle \xrightarrow{\tau}^* s_2 \parallel \langle n'_e, [r_e \to 0]v'\rangle$$

As, $(s \parallel \langle n', v'\rangle, u) \models \phi_1 \wedge \phi_2$, it follows, by Proposition 4.6, that

$$(s_2 \parallel \langle n'_e, [r_e \to 0]v'\rangle, u) \models \varphi$$

By definition of the environment $\rho$, we obtain that $(s_2, [r_e \to 0]v' : u) \in [\![\phi/n'_e]\!]\rho$. We may therefore conclude that

$$(s_1, v' : u) \in [\![g_e \Rightarrow [\bar{b}](r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))]\!]\rho$$

which was to be shown. The proof for the other cases for $\psi$ follows very similar lines, and is left to the reader.
- CASE $\varphi \equiv g \vee \phi$. The proof is the same as the previous one for $\varphi \equiv \phi_1 \wedge \phi_2$, and is therefore omitted.
- CASE $\varphi \equiv [a]\phi$. We prove that

$$(s, v' : u) \in \left[\!\left[ \begin{array}{c} [a](\phi/n') \wedge \bigwedge_{e \in E(n',a)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi/n'_e) \\ \wedge \\ \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ ([a]\phi)/n'_e) \\ \wedge \\ \bigwedge_{b \in \mathsf{Act}} \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\bar{b}]\, (r_e \ \underline{\mathtt{in}}\ ([a]/\phi)/n'_e)) \end{array} \right]\!\right] \rho$$

To this end, assume that $s \xrightarrow{\tau}^* s_1$. We aim at proving that $\langle s_1, v' : u\rangle \in [\![\psi]\!]\rho$, for every formula $\psi$ in the set of formulae:

$$\{[a](\phi/n')\} \cup \bigcup_{e \in E(n',a)} \{g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi/n'_e\} \cup \bigcup_{e \in E(n',\tau)} \{g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ ([a]\phi)/n'_e\}$$
$$\cup \bigcup_{b \in \mathsf{Act}} \bigcup_{e \in E(n',b)} \{g_e \Rightarrow [\bar{b}]\, (r_e \ \underline{\mathtt{in}}\ ([a]/\phi)/n'_e)\}$$

We present the details of the proof for the case $\psi \equiv g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi/n'_e$ with $e \in E(n', a)$. The other cases are dealt with in similar fashion, and are left to the reader.
Assume that $e \in E(n', a)$ and $g_e(v')$ is true. Then, the following holds:

$$s \parallel \langle n', v'\rangle \overset{a}{\Longrightarrow} s_1 \parallel \langle n'_e, v'_1\rangle$$

where $v'_1 = [r_e \to 0]v'$. Since $(s \parallel \langle n', v'\rangle, u') \models \varphi$, we infer that $(s_1 \parallel \langle n'_e, v'_1\rangle, u) \models \phi$. By the definition of the environment $\rho$, it follows that $(s_1, v'_1 : u) \in [\![\phi/n'_e]\!]\rho$. We may therefore conclude that $(s_1, v' : u) \in [\![r_e \ \underline{\mathtt{in}}\ (\phi/n'_e)]\!]\rho$, which was to be shown.

- CASE $\varphi \equiv x \ \underline{\mathtt{in}}\ \phi$. We aim at showing that

$$(s, v' : u) \in \left[\!\!\left[ \begin{array}{c} x \ \underline{\mathtt{in}}\ (\phi/n') \\ \wedge \\ \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (x \ \underline{\mathtt{in}}\ \phi)/n'_e) \\ \wedge \\ \bigwedge_{b \in \mathsf{Act}} \bigwedge_{e \in E(n',b)} (g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (x \ \underline{\mathtt{in}}\ \phi)/n'_e)) \end{array} \right]\!\!\right] \rho$$

The case that we will study is the case $x \ \underline{\mathtt{in}}\ (\phi/n')$ because for the other, it is the same as before. Assume that $(s, v' : u) \xrightarrow{\tau}^* (s_1, v' : u)$. Then by synchronizing with the second component, the following holds : $(s, u) \parallel (\langle n', v'\rangle, u) \xrightarrow{\tau}^* (s_1, u) \parallel (\langle n', v'\rangle, u)$. By assumption, $(s_1, [x \leftarrow 0]u) \parallel (\langle n', v'\rangle, [x \leftarrow 0]u) \models \phi$. Then $(s_1, v' : [x \leftarrow 0]u), \phi \in [\![\phi/n']\!]\rho$. Thus $(s, v' : u) \in [\![x \ \underline{\mathtt{in}}\ (\phi/n')]\!]\rho$. So, we are done.
- CASE $\varphi = \forall\!\!\!\forall_S \phi$. We distinguish two sub-cases, depending on whether $S \cap \mathcal{U}(n') = \emptyset$ or not.
  * CASE: $S \cap \mathcal{U}(n') = \emptyset$. We prove that if $s \overset{\epsilon(d)}{\Longrightarrow}_{S \cup \overline{\mathcal{U}(n')}} s_1$ with $d \in \mathbb{R}_{\geq 0}$, then

$$(s_1, v' + d : u + d) \in \left[\!\!\left[ \begin{array}{c} \phi/n' \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e)) \\ \wedge \\ \bigwedge_{b \in \mathsf{Act}} \bigwedge_{e \in E(n',b)} (g_2 \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))) \end{array} \right]\!\!\right] \rho$$

  To this end, we prove that if $s_1 \xrightarrow{\tau}^* s_2$ then $(s_2, v' + d : u + d) \in [\![\psi]\!]\rho$, for every conjunct $\psi$ of the right-hand side of the defining equation for $\varphi/n'$.
  The case which is really interesting and different from the others that we have previously considered is

$$(s_2, v' + d : u + d) \in [\![\phi/n']\!]\rho$$

  because the proofs for the other cases are those used for the case $\varphi \equiv \phi_1 \wedge \phi_2$. To handle this case, we reason as follows. Since $S \cap \mathcal{U}(n') = \emptyset$, we may infer that $\langle n', v'\rangle \xrightarrow{\epsilon(d)}_S \langle n', v' + d\rangle$. Synchronizing this delay transition from state $\langle n', v'\rangle$ with the computation

$$s \overset{\epsilon(d)}{\Longrightarrow}_{S \cup \overline{\mathcal{U}(n')}} s_1 \xrightarrow{\tau}^* s_2$$

  yields that

$$s \parallel \langle n', v'\rangle \overset{\epsilon(d)}{\Longrightarrow}_S s_2 \parallel \langle n', v' + d)\rangle$$

  because the first component is unable to synchronize with the second on urgent actions during the delays. By hypothesis, $(s_2 \parallel \langle n', v' + d\rangle, u + d) \models \phi$. By the definition of the environment $\rho$, we obtain that $(s_2, v' + d : u + d) \in [\![\phi/n']\!]\rho$, which was to be shown.
  * CASE: $S \cap \mathcal{U}(n') \neq \emptyset$. The details of the proof are the same for the case $\varphi \equiv \phi_1 \wedge \phi_2$.
- CASE $\varphi \equiv \mathtt{max}(X, \phi)$. Assume that $s \xrightarrow{\tau}^* s_1$. We prove that

$$(s_1, v' : u) \in \left[\!\!\left[ \begin{array}{c} (\phi\{\mathtt{max}(X, \phi)/X\})/n' \wedge \bigwedge_{e \in E(n',\tau)} (g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e)) \\ \wedge \\ \bigwedge_{b \in \mathsf{Act}} \bigwedge_{e \in E(n',b)} (g_e \ \underline{\mathtt{in}}\ [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))) \end{array} \right]\!\!\right] \rho$$

To this end, let us assume that $s_1 \xrightarrow{\tau}^* s_2$. We aim at proving that $(s_2, v' : u) \in [\![\psi]\!]\rho$ when $\psi$ is any of the formulae in the set

$$\{(\phi\{\mathtt{max}(X, \phi)/X\})/n'\} \cup \bigcup_{e \in E(n',\tau)} \{g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e)\} \cup \bigcup_{b \in \mathsf{Act}} \bigcup_{e \in E(n',b)} \{g_e \Rightarrow [\overline{b}]\,(r_e \ \underline{\mathtt{in}}\ (\varphi/n'_e))\}$$

Let us consider the formula $\psi \equiv (\phi\{\mathtt{max}(X,\phi)/X\})/n'$. Since $s \xrightarrow{\tau}^{*} s_2$, it follows that

$$s \parallel \langle n', v'\rangle \xrightarrow{\tau}^{*} s_2 \parallel \langle n', v'\rangle$$

By definition of the satisfaction relation for $\varphi$, we have

$$(s_2 \parallel \langle n', v'\rangle, u) \models \phi\{\mathtt{max}(X,\phi)/X\}$$

By the definition of the environment $\rho$, we may now infer that

$$(s_2, v' : u) \in [\![(\phi\{\mathtt{max}(X,\phi)/X\})/n']\!]\rho$$

Consequently, $(s_2, v' : u) \in [\![(\phi\{\mathtt{max}(X,\phi)/X\})/n']\!]\rho$, that is what we wanted to prove.
For the other possible forms $\psi$ may take, the proof is similar to the one for $\varphi \equiv \phi_1 \wedge \phi_2$ because of Proposition 4.6. $\square$

The proof of the 'only if' implication is now complete.

This completes the proof of the theorem. $\square$

## C Proofs omitted from Section 6

**Theorem 6.4.** *Let $A$ and $B$ be two timed automata. Assume that $B$ is deterministic and $\tau$-free. Then, for every node $n$ of $A$ and $m$ of $B$, and valuations $v, w$ for the clocks in $A$ and $B$, respectively,*

$$\langle n, v\rangle \preceq_w \langle m, w\rangle \;\; \textit{iff} \;\; (\langle n, v\rangle, w) \models \phi(m) \;\; .$$

*Proof.* We prove the two implications separately.

- 'IF' IMPLICATION: We show that if $(\langle n, v\rangle, w) \models \phi(m)$, then $\langle n, v\rangle \preceq_w \langle m, w\rangle$. To this end, it is sufficient to prove that the relation:

$$\mathcal{R} = \{(\langle n, v\rangle, \langle m, w\rangle) \mid m \in \mathtt{Nodes}(B),\; n \in \mathtt{Nodes}(A) \text{ and } (\langle n, v\rangle, w) \models \phi(m)\}$$

  satisfies the defining clauses of $\preceq_w$. Assume therefore that $(\langle n, v\rangle, \langle m, w\rangle) \in \mathcal{R}$. We check that the transfer properties of $\preceq_w$ are met by $\mathcal{R}$.

  1. (ACTION TRANSITIONS) Assume that $\langle n, v\rangle \xRightarrow{a} \langle n', v'\rangle$. We show that $\langle m, w\rangle \xrightarrow{a} \langle m', w'\rangle$ for some node $m'$ and valuation $w'$ such that $(\langle n', v'\rangle, \langle m', w'\rangle) \in \mathcal{R}$. Note, first of all, that, as $(\langle n, v\rangle, w) \models \phi(m)$, we may infer that

$$(\langle n, v\rangle, w) \models [a]\Big(\textsc{Enabled}(m,a) \wedge \textsc{Match}(m,a)\Big)$$

     Since $\langle n, v\rangle \xRightarrow{a} \langle n', v'\rangle$, it follows that

$$(\langle n', v'\rangle, w) \models \textsc{Enabled}(m,a) \quad \text{and} \tag{12}$$

$$(\langle n', v'\rangle, w) \models \textsc{Match}(m,a) \tag{13}$$

     By (12), the clock valuation $w$ enables at least one of the edges in $E(m,a)$. Call this edge $e$. (In fact, as $B$ is deterministic, $e$ is the only edge in $E(m,a)$ that is enabled by $w$.) Since $\langle m, w\rangle \xrightarrow{a} \langle m_e, [r_e \to 0]w\rangle$, we are left to show that $(\langle n', v'\rangle, \langle m_e, [r_e \to 0]w\rangle) \in \mathcal{R}$. By (13) and $g_e(w)$, we obtain that

$$(\langle n', v'\rangle, w) \models r_e \;\underline{\mathtt{in}}\; \phi(m_e)$$

     This yields that

$$(\langle n', v'\rangle, [r_e \to 0]w) \models \phi(m_e) \tag{14}$$

     By (14) and the definition of $\mathcal{R}$, we finally conclude that $(\langle n', v'\rangle, \langle m_e, [r_e \to 0]w\rangle) \in \mathcal{R}$, which was to be shown.

2. (Delay Transitions) Assume that $\langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_S \langle n', v'\rangle$ for some $d \geq 0$ and set of urgent actions $S$. We show that $\langle m, w\rangle \stackrel{\epsilon(d)}{\longrightarrow}_S \langle m, w+d\rangle$ and that $(\langle n', v'\rangle, \langle m, w+d\rangle) \in \mathcal{R}$. This we do by distinguishing two cases, depending on whether $d = 0$ or not.
   - Case $d = 0$. In this case $\langle n, v\rangle \stackrel{\tau}{\longrightarrow}^* \langle n', v'\rangle$, since the relation $\stackrel{\epsilon(0)}{\Longrightarrow}_S$ coincides with $\stackrel{\tau}{\longrightarrow}^*$. Proposition 4.6 now yields that $(\langle n', v'\rangle, w) \models \phi(m)$. By the definition of the relation $\mathcal{R}$, we may finally conclude that $(\langle n', v'\rangle, \langle m, w\rangle) \in \mathcal{R}$, which was to be shown.
   - Case $d > 0$. We begin by showing that $\langle m, w\rangle \stackrel{\epsilon(d)}{\longrightarrow}_S \langle m, w+d\rangle$. Note, first of all, that $\langle m, w\rangle \stackrel{\epsilon(d)}{\longrightarrow} \langle m, w+d\rangle$ by the definition of delay transitions for states of timed automata. We now argue that $\langle m, w\rangle \stackrel{\epsilon(d)}{\longrightarrow}_S \langle m, w+d\rangle$ also holds, *i.e.*, that $\langle m, w+d'\rangle \stackrel{a}{\not\longrightarrow}$, for every $a \in S$. To this end, assume, towards a contradiction, that there is an $a \in S$ such that $\langle m, w\rangle \stackrel{a}{\longrightarrow}$. Since $a$ is urgent, this means that there is an edge $e \in E(m, a)$ whose guard $g_e$ is a tautology. Thus $\langle m, w\rangle \stackrel{a}{\longrightarrow}$. As we are assuming that $(\langle n, v\rangle, w) \models \phi(m)$ and the valuation $w$ satisfies $g_e$, it follows that $(\langle n, v\rangle, w) \models \langle a\rangle\mathtt{tt}$. This yields that every state that is reachable from $\langle n, v\rangle$ by performing a sequence of $\tau$-labelled transitions affords an $a$-labelled transition. However, as $d > 0$, this contradicts our assumption that $\langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_S \langle n', v'\rangle$. It therefore follows that $\langle m, w\rangle \stackrel{\epsilon(d)}{\longrightarrow}_S \langle m, w+d\rangle$, which was to be shown.
     We now proceed to argue that $(\langle n', v'\rangle, \langle m, w+d\rangle) \in \mathcal{R}$. By the definition of $\mathcal{R}$, it is sufficient to show that $(\langle n', v'\rangle, w+d) \models \phi(m)$. However, this follows because
     * $\phi(m)$ is logically equivalent to $\forall\!\!\!\forall_\emptyset \phi(m)$ (Lemma 4.8(2)), and
     * $\langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_\emptyset \langle n', v'\rangle$, as $\langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_S \langle n', v'\rangle$.

   This completes the proof for this transfer property.
3. (Readiness) Assume that $\langle m, w\rangle \stackrel{a}{\longrightarrow}$ for some urgent action $a$. We prove that $\langle n, v\rangle \stackrel{a}{\longrightarrow}$ also holds. To this end, note that, as $\langle m, w\rangle \stackrel{a}{\longrightarrow}$, there is an edge $e \in E(m, a)$, whose guard $g_e$ is a tautology, as $a$ is urgent. Since $(\langle n, v\rangle, w) \models \phi(m)$ and $w$ satisfies $g_e$, it follows that $(\langle n, v\rangle, w) \models \langle a\rangle\mathtt{tt}$. We may therefore conclude that $\langle n, v\rangle \stackrel{a}{\longrightarrow}$, which was to be shown.

We have thus shown that the relation $\mathcal{R}$ satisfies the defining clauses of $\preceq_w$, as required.

– 'Only If' Implication: We aim at showing that $\langle n, v\rangle \preceq_w \langle m, w\rangle$ implies $(\langle n, v\rangle, w) \models \phi(m)$. To this end, it is sufficient to prove that the environment $\rho$ defined thus, for every node $l$ of $B$,

$$[\![\phi(l)]\!]\rho \stackrel{\text{def}}{=} \{(\langle n', v'\rangle, w') \mid \langle n', v'\rangle \preceq_w \langle l, w'\rangle\}$$

is a post-fixed point of the monotonic functional on environments induced by (5). This we now proceed to show.

Assume that $(\langle n, v\rangle, w) \in [\![\phi(m)]\!]\rho$, *i.e.*, that $\langle n, v\rangle \preceq_w \langle m, w\rangle$. We wish to argue that the extended state $(\langle n, v\rangle, w)$ is contained in the interpretation of the right-hand side of (5) with respect to the environment $\rho$. Suppose, to this end, that $\langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_\emptyset \langle n', v'\rangle$ for some $d \geq 0$. We prove that

$$(\langle n', v'\rangle, w+d) \in [\![ \bigwedge_{a \in \mathsf{Act}} [a]\Big(\textsc{Enabled}(m, a) \wedge \textsc{Match}(m, a)\Big) \wedge \bigwedge_{a \in \mathcal{U}} \textsc{Out}(m, a)]\!]\rho \qquad (15)$$

To establish (15), in light of Lemma 4.8(3), it is sufficient to argue that if $\langle n', v'\rangle \stackrel{\tau}{\longrightarrow}^* \langle n'', v''\rangle$ then, for every action $a$,

$$(\langle n'', v''\rangle, w+d) \in [\![[a]\textsc{Enabled}(m, a)]\!]\rho \qquad (16)$$
$$(\langle n'', v''\rangle, w+d) \in [\![[a]\textsc{Match}(m, a)]\!]\rho \quad \text{and} \qquad (17)$$
$$(\langle n'', v''\rangle, w+d) \in [\![\textsc{Out}(n, a)]\!]\rho \quad \text{if } a \text{ is urgent} \qquad (18)$$

We consider each of these statements in turn.

- *Proof of* (16). Assume that $\langle n'', v''\rangle \stackrel{a}{\Longrightarrow} \langle \bar{n}, \bar{v}\rangle$. We prove that $(\langle \bar{n}, \bar{v}\rangle, w + d)$ is contained in $[\![\textsc{Enabled}(m, a)]\!]\rho$. To see that this does hold, recall that

$$\begin{array}{l} \langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\emptyset} \langle n', v'\rangle \stackrel{\tau}{\longrightarrow}^{*} \langle n'', v''\rangle \stackrel{a}{\Longrightarrow} \langle \bar{n}, \bar{v}\rangle \\ \quad \preceq_w \\ \langle m, w\rangle \end{array}$$

Using the definition of the preorder $\preceq_w$ and the fact that the timed automaton $B$ is $\tau$-free, it follows that the above diagram can be completed, for some node $m'$ and valuation $w'$, as in Figure 10. Since $\langle m, w + d\rangle \stackrel{a}{\longrightarrow} \langle m', w'\rangle$, there is an edge $e$ contained in $E(m, a)$

$$\begin{array}{ccccccc} \langle n, v\rangle & \stackrel{\epsilon(d)}{\Longrightarrow}_{\emptyset} & \langle n', v'\rangle & \stackrel{\tau}{\longrightarrow}^{*} & \langle n'', v''\rangle & \stackrel{a}{\Longrightarrow} & \langle \bar{n}, \bar{v}\rangle \\ \preceq_w & & \preceq_w & & \preceq_w & & \preceq_w \\ \langle m, w\rangle & \stackrel{\epsilon(d)}{\longrightarrow}_{\emptyset} & \langle m, w + d\rangle & \equiv & \langle m, w + d\rangle & \stackrel{a}{\longrightarrow} & \langle m', w'\rangle \end{array}$$

**Figure 10.** Matching Transitions

whose guard $g_e$ is satisfied by the valuation $w + d$. (In fact, as $B$ is deterministic, this edge is unique.) This yields that $(\langle \bar{n}, \bar{v}\rangle, w + d)$ is contained in the collection of extended states $[\![\textsc{Enabled}(m, a)]\!]\rho$, and thus that (16) holds.
- *Proof of* (17). To see that (17) also holds, recall that, as previously observed, $e$ is the only edge in $E(m, a)$ whose guard is enabled by the valuation $w + d$. Hence, in the diagram in Figure 10, it must be the case that $m' = m_e$ and $w' = [r_e \to 0](w + d)$. Since $\langle \bar{n}, \bar{v}\rangle \preceq_w \langle m_e, [r_e \to 0](w+d)\rangle$, we may infer that the extended state $(\langle \bar{n}, \bar{v}\rangle, [r_e \to 0](w+d))$ is contained in $[\![\phi(m_e)]\!]\rho$. This yields that

$$(\langle \bar{n}, \bar{v}\rangle, w + d) \in [\![g_e \Rightarrow r_e \ \underline{\mathtt{in}}\ \phi(n_e)]\!]\rho$$

Claim (17) now follows immediately because $e$ is the only edge which is enabled by the valuation $w + d$.
- *Proof of* (18). The only interesting thing to check is that if the valuation $w + d$ satisfies the guard of some edge $e \in E(m, a)$ ($a$ urgent), then every state $\langle \bar{n}, \bar{v}\rangle$ that can be reached from $\langle n'', v''\rangle$ via a (possibly empty) sequence of $\tau$-labelled transitions has an outgoing $a$-labelled transition. Recall that, since $a$ is urgent, the guard $g_e$ is a tautology. Assume therefore that $\langle n'', v''\rangle \stackrel{\tau}{\longrightarrow}^{*} \langle \bar{n}, \bar{v}\rangle$. We now argue that $\langle \bar{n}, \bar{v}\rangle \stackrel{a}{\longrightarrow}$. Recall that

$$\begin{array}{l} \langle n, v\rangle \stackrel{\epsilon(d)}{\Longrightarrow}_{\emptyset} \langle n', v'\rangle \stackrel{\tau}{\longrightarrow}^{*} \langle n'', v''\rangle \stackrel{\tau}{\longrightarrow}^{*} \langle \bar{n}, \bar{v}\rangle \\ \quad \preceq_w \\ \langle m, w\rangle \end{array}$$

Using the definition of the preorder $\preceq_w$ and the fact that the timed automaton $B$ is $\tau$-free, it follows that the above diagram can be completed thus:

$$\begin{array}{ccccccc} \langle n, v\rangle & \stackrel{\epsilon(d)}{\Longrightarrow}_{\emptyset} & \langle n', v'\rangle & \stackrel{\tau}{\longrightarrow}^{*} & \langle n'', v''\rangle & \stackrel{\tau}{\longrightarrow}^{*} & \langle \bar{n}, \bar{v}\rangle \\ \preceq_w & & \preceq_w & & \preceq_w & & \preceq_w \\ \langle m, w\rangle & \stackrel{\epsilon(d)}{\longrightarrow}_{\emptyset} & \langle m, w + d\rangle & \equiv & \langle m, w + d\rangle & \equiv & \langle m, w + d\rangle \end{array}$$

Since valuation $w + d$ satisfies the guard of $e \in E(m, a)$ and $g_e$ is a tautology, we may infer that $\langle m, w + d\rangle \stackrel{a}{\longrightarrow}$. Using the urgency of $a$ and $\langle \bar{n}, \bar{v}\rangle \preceq_w \langle m, w + d\rangle$, we may now finally conclude that $\langle \bar{n}, \bar{v}\rangle \stackrel{a}{\longrightarrow}$, which was to be shown.

This completes the proof of (15), and thus of the "only if" implication.

The proof of the theorem is finally complete. $\square$

**Proposition 6.6.** *For every formula $\varphi \in L_{\forall S}$ and every valuation $w$ for the clocks in $K$,*

*if $(a.(b+c)+a.(b+d),w) \models \varphi$, then $(a.(b+c+d),w) \models \varphi$.*

In the proof of the above result, we shall make use of the following auxiliary lemma.

**Lemma 6.9.** *For every formula $\varphi \in L_{\forall S}$ and every clock valuation $w$, if $(b+c,w) \models \varphi$ and $(b+d,w) \models \varphi$, then $(b+c+d,w) \models \varphi$*

*Proof.* Consider the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \models \ \cup \{((b+c+d,w),\varphi) \mid (b+c,w) \models \varphi \text{ and } (b+d,w) \models \varphi\} \ .$$

We prove that $\mathcal{R}$ satisfies the defining clauses of $\models$ (see Table 4). Assume therefore that

$$(\langle b+c+d,w\rangle,\varphi) \in \mathcal{R} \ .$$

By the definition of $\mathcal{R}$, we have that $(b+c,w) \models \varphi$ and $(b+d,w) \models \varphi$. We proceed by a case analysis on the form $\varphi$ may take, and only consider a few selected cases.

- CASE $\varphi \equiv \langle a'\rangle\mathtt{tt}$, for some action $a'$. Vacuous because $a'$ is urgent, but none of $b,c,d$ is.
- CASE $\varphi \equiv [a']\phi$, for some action $a'$. If $a' \notin \{b,c,d\}$, then the relevant defining clause of $\models$ is vacuously met. Assume thus that $a' \in \{b,c,d\}$ and $b+c+d \xrightarrow{a'} s$. We argue that $((s,w),\phi) \in \mathcal{R}$. First of all, note that $s \equiv \mathrm{NIL}$, where NIL denotes the stopped process of TCCS. Now, since $(b+c,w) \models \varphi$ and $(b+d,w) \models \varphi$, we infer that $(\mathrm{NIL},w) \models \phi$. The claim now follows immediately because $\models$ is included in $\mathcal{R}$.
- CASE $\varphi \equiv \forall\!\!\!\forall_S\phi$. Assume that $b+c+d \xrightarrow{\epsilon(e)}_S s$ for some $e \geq 0$ and state $s$. Note, first of all, that $s \equiv b+c+d$. Since none of $b,c,d$ is contained in $S$, infer that $(b+c,w) \models \varphi$ and $b+c \xrightarrow{\epsilon(e)}_S b+c$. This yields that $(b+c,w+e) \models \phi$. A similar reasoning yields that $(b+d,w+e) \models \phi$. Using the definition of $\mathcal{R}$, we may now conclude that $((b+c+d,w+e),\phi) \in \mathcal{R}$, which was to be shown.

The claim now follows because $\models$ coincides with $\mathcal{R}$, since $\models$ is the largest relation satisfying the clauses in Table 4. $\square$

We are now in a position to prove Proposition 6.6.

*Proof of Proposition 6.6:* Consider the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \models \ \cup \{((a.(b+c+d),w),\varphi) \mid (a.(b+c)+a.(b+d),w) \models \varphi\} \ .$$

We prove that $\mathcal{R}$ satisfies the defining clauses of $\models$ (see Table 4). Assume therefore that

$$((a.(b+c+d),w),\varphi) \in \mathcal{R} \ .$$

By the definition of $\mathcal{R}$, we have that $(a.(b+c)+a.(b+d),w) \models \varphi$. We proceed by a case analysis on the form $\varphi$ may take, and only consider two selected cases.

- CASE $\varphi \equiv [a']\phi$, for some action $a'$. If $a' \neq a$, then the relevant defining clause of $\models$ is vacuously met. Assume thus that $a' = a$ and $a.(b+c+d) \xrightarrow{a'} s$. We argue that $((s,w),\phi) \in \mathcal{R}$. First of all, note that $s \equiv b+c+d$. Next, since $(a.(b+c)+a.(b+d),w) \models \varphi$, we infer that $(b+c,w) \models \phi$ and $(b+d,w) \models \phi$. Lemma 6.9 yields that $(b+c+d,w) \models \phi$. The claim now follows immediately because $\models$ is included in $\mathcal{R}$.
- CASE $\varphi \equiv \forall\!\!\!\forall_S\phi$. Assume that $a.(b+c+d) \xrightarrow{\epsilon(e)}_S s$ for some $e \geq 0$ and state $s$. Note, first of all, that $s \equiv a.(b+c+d)$. We wish to argue that $((a.(b+c+d),w+e),\phi) \in \mathcal{R}$. Since $a.(b+c+d) \xrightarrow{\epsilon(e)}_S a.(b+c+d)$ and $a \notin S$ ($a$ is not urgent), it holds that cases $a.(b+c)+a.(b+d) \xrightarrow{\epsilon(e)}_S a.(b+c)+a.(b+d)$. Note now that $(a.(b+c)+a.(b+d),w+e) \models \phi$. Using the definition of $\mathcal{R}$, we may finally conclude that $((a.(b+c+d),w+e),\phi) \in \mathcal{R}$, which was to be shown. $\square$