Axiomatising Timed Automata *

Huimin Lin †
Laboratory for Computer Science
Inst. of Software, Chinese Academy of Sciences
Email: lhm@ox.ios.ac.cn

Wang Yi
Dept. of Computer Systems
Uppsala University
Email: yi@csd.uu.se

Abstract

Timed automata has been developed as a basic semantic model for real time systems. Its algorithmic aspects for automated analysis have been well studied. But so far there is still no satisfactory algebraic theory to allow the derivation of semantical equivalence of automata by purely syntactical manipulation. The aim of this paper is to provide such a theory. We present an inference system of timed bisimulation equivalence for timed automata based on a CCS-style regular language for describing timed automata. It consists of the standard monoid laws for bisimulation and a set of inference rules. The judgments of the proof system are *conditional equations* of the form $\phi \rhd t = u$ where ϕ is a clock constraint and t, u are terms denoting timed automata. The inference system is shown to be sound and complete for timed bisimulation. The proof of the completeness result relies on the notion of *symbolic timed bisimulation*, adapted from the work on value–passing processes.

1 Introduction

The last decade has seen a growing interest in extending various concurrency theories with timing constructs so that real-time aspects of concurrent systems can be modeled. Among them timed automata [AD94] has stood out as a fundamental model for real-timed systems.

A timed automaton is a finite automaton extended with a finite set of real-valued *clock variables*. A node of a timed automata is associated with an *invariant* constraint on the clock variables, while an edge is decorated with a clock constraint, an action label, and a subset of clocks to be reset after the transition. At each node a timed automaton may perform two kinds of transitions: it may let time pass for any amount (a delay

*An extended abstract of this paper appeared in *Proc. FST&TCS 2000: Foundations of Software Technology and Theoretical Computer Science*, pp.277-289, LNCS 1974. December 2000.

†Supported by National Science Foundation of China

transition), as long as the invariant is satisfied, or choose an edge whose constraint is met, make the move, reset the relevant clocks to zero, and arrive at the target node (an action transition). Although a timed automaton has only finite number of nodes, due to (real-valued) delay transitions it typically exhibits infinite-state behaviour. Two timed automata are *timed bisimilar* if they can match each other's action transitions as well as delay transitions, and their residuals remain timed bisimilar. The explicit presence of clock variables and resetting, features that mainly associated with the so-called "imperative languages", distinguishes timed automata from process calculi such as CCS, CSP and their timed extensions which are "applicative" in nature and therefore more amenable to axiomatisation. By now most theoretical aspects of timed automata have been well studied, but they still lack a satisfactory algebraic theory to allow the derivation of semantically correct assertions by purely syntactical manipulation.

The aim of this paper is to provide such a theory. We shall develop a complete axiomatisation for timed automata, in the form of an inference system, in which the equalities between pairs of timed automata that are timed bisimilar can be derived. To this end we first propose a language, in CCS style, equipping it with a symbolic transitional semantics in such a way that each term in the language denotes a timed automaton. The language has a conditional construct $\phi{\rightarrow}t$, read "if ϕ then t", an action prefixing $a(\mathbf{x}).t$, meaning "perform the action a, reset the clocks in $\mathbf{x}$ to zero, then behave like t", an invariant $\{\phi\}t$ which puts the delay behaviour of t under the restriction imposed by ϕ, and a recursion $\mathbf{fix}Xt$ which allows infinite behaviours to be described. The proof system consists of a set of inference rules and the standard monoid laws for bisimulation. Roughly speaking the monoid laws characterize bisimulation, while the inference rules deal with specific constructs in the language.

Because terms usually contain free clock variables, two terms may be equal only when the values of the clock variables in them satisfy certain condition. Reflecting this the judgments of the inference system are *conditional equations* of the form

$$\phi \rhd t = u$$

where ϕ is a time constraint and t, u are terms. Intuitively it means: t and u are timed bisimilar over clock evaluations satisfying ϕ. To give a taste of the inference system, here is the rule for the "if - then" construction:

$$\text{GUARD} \quad \frac{\phi \wedge \psi \rhd t = u \quad \phi \wedge \neg\psi \rhd \mathbf{0} = u}{\phi \rhd \psi{\rightarrow}t = u}$$

It performs a case analysis on the constraint ψ: $\psi{\rightarrow}t$ behaves like t when ψ is true, and like the inactive process $\mathbf{0}$ otherwise. Note that the guarding constraint ψ of $\psi{\rightarrow}t$ in the conclusion is *part of the object language* describing timed automata, while in the premise it is shifted to the condition part of the judgment in our *meta language* for reasoning about timed automata.

A crucial rule, as might be expected, is the one for action prefixing:

$$\text{ACTION} \quad \frac{\phi{\downarrow_{\mathbf{x}}}{\Uparrow} \rhd t = u}{\phi \rhd a(\mathbf{x}).t = a(\mathbf{x}).u}$$

Here $\downarrow_{\mathbf{x}}$ and $\Uparrow$ are postfixing operations on clock constraints. $\phi\downarrow_{\mathbf{x}}\Uparrow$ is a clock constraint obtained from ϕ by first setting the clocks in $\mathbf{x}$ to zero (operator $\downarrow_{\mathbf{x}}$), then removing up-bounds on all clocks of ϕ (operator $\Uparrow$). Readers familiar with Hoare Logic may have noticed some similarity between this rule and the triple dealing with assignment there:

$$\{P[e/x]\}\ x := e\ \{P\}$$

But here the operator $\downarrow_{\mathbf{x}}$ is slightly more complicated than substitution with zero, because clocks are required to increase uniformly. Also we need $\Uparrow$ to allow time to pass indefinitely.

A standard way to reasoning with recursion is to use, in addition to the usual rule for folding/unfolding recursions, the following *unique fixpoint induction*:

$$\text{UFI}\quad \frac{t = u[t/X]}{t = \mathbf{fix}Xu}\quad X \text{ guarded in } u$$

This rule was adopted in [Mil84] for a complete axiomatisation of bisimulation equivalence for regular pure-CCS. Here we use it in a quite different context: terms in our setting normally contain clock variables, namely they are *open terms*. In spite of this, it turns out that this rule is still sound and sufficient for a complete axiomatisation of regular timed behaviour, though the proof is more complicated than in the pure calculi.

The completeness theorem states: if t and u are timed bisimilar then $t = u$ can be derived in the proof system. Because of delay transitions comparing a pair of terms for timed bisimulation involves matching infinite number of residuals. Such infinity makes it difficult to directly use timed bisimulation in constructing the completeness proof. To overcome this drawback we introduce the notion of *symbolic timed bisimulation*, $t \sim^{\phi} u$, which encodes the matching of delay transitions as the *upward closure* property of indexing constraint ϕ, and only requires explicit matching of action transitions which are finite in number for a pair of terms. It captures timed bisimulation in the sense that $t \sim^{\phi} u$ if and only if $t\rho$ and $u\rho$ are timed bisimilar for any clock evaluation ρ satisfying ϕ.

With symbolic timed bisimulation the completeness theorem can be reformulated as: $t \sim^{\phi} u$ implies $\vdash \phi \rhd t = u$. Following [Mil84], to show that the inference system is complete, we first transform t and u into *standard equation sets* which are the syntactical representations of timed automata. We then construct a product equation set out of the two and prove that t and u both satisfy this new equation set, by exploiting the assumption that t and u are symbolically timed bisimilar. Due to the presence of clock variables the notion of satisfiability is relative to a set of clock constraints. Finally we show that, using UFI, if two terms satisfy the same set of standard equations then they are provably equal.

The rest of the paper is organised as follows: In the next section we first recall the definition of timed automata, then propose a language to describe them. Section 3 introduces symbolic timed bisimulation. The inference system is presented and its soundness discussed in Section 4. Section 5 is devoted to proving the completeness of

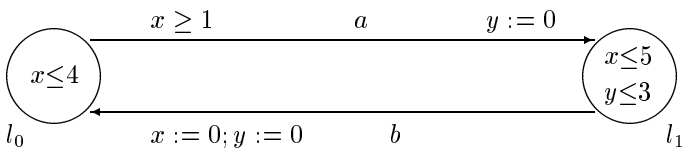


Figure 1: A Timed Automaton.

the proof system. The paper concludes with Section 6 where related work is also briefly discussed.

2 A Language for Timed Automata

2.1 Timed Automata

A timed automaton is a standard finite-state automaton extended with a finite collection of real-valued clocks. In a timed automaton, each node is associated with an *invariant*, while a transition is labelled with a *guard* (a constraint on clocks), a *synchronisation action*, and a *clock reset* (a subset of clocks to be reset). Intuitively, a timed automaton starts execution with all clocks initialized to zero. The automaton can stay at a node, as long as the invariant of the node is satisfied, with all clocks increasing at the same rate. A transition can be taken if the values of the clocks fulfill the guard. By taking the transition, all clocks in the clock reset are set to zero, while the others keep their values. Semantically, a state of an automaton is a pair consisting of a control node and a *clock valuation*, i.e. the current setting of the clocks. Transitions in the semantic interpretation are either labelled with a synchronisation action (if it is an instantaneous switch from the current node to another) or with a non-negative real number, i.e. a time delay (if the automaton stays within a node letting time pass).

Consider the timed automaton of Figure 1. It has two control nodes l_0 and l_1 and two clocks x and y. A *state* of the automaton is of the form $(l, \langle s, t \rangle)$, where l is a control node and s and t are non–negative reals giving the values of x and y. Assuming that the automaton starts to operate in the state $(l_0, \langle 0, 0 \rangle)$, it may stay in node l_0 for any amount of time, as long as the invariant $x \leq 4$ at l_0 is satisfied. During this time the values of x and y increase uniformly, at the same rate. Thus from the initial state, all states of the form $(l_0, \langle t, t \rangle)$ with $0 \leq t \leq 4$ are reachable, but only at the states $(l_0, \langle t, t \rangle)$, where $t \geq 1$, the edge from l_0 to l_1 is enabled. When following the edge from l_0 to l_1 the action a is performed to synchronize with the environment and the clock y is reset to 0 leading to states of the form $(l_1, \langle t, 0 \rangle)$ where $t \geq 1$.

For the formal definition, we assume a finite set $\mathcal{A}$ for synchronization actions and a finite set $\mathcal{C}$ for real-valued clock variables. We use a, b etc. to range over $\mathcal{A}$ and x, y *etc.* to range over $\mathcal{C}$. We use $\mathcal{B}(\mathcal{C})$, ranged over by ϕ, ψ *etc.*, to denote the set of conjunctive formulas of atomic constraints in the form $x_i \bowtie m$ or $x_i - x_j \bowtie n$, where

$x_i, x_j \in \mathcal{C}$, $\bowtie \in \{\leq, <, \geq, >\}$ and m, n are natural numbers. The elements of $\mathcal{B}(\mathcal{C})$ are called *clock constraints*. The constants tt and ff are used to denote, respectively, the atomic constraints which are constantly true and false.

Definition 1 *A timed automaton over actions $\mathcal{A}$ and clocks $\mathcal{C}$ is a tuple $\langle N, l_0, I, E\rangle$ where*

- *N is a finite set of nodes,*
- *$l_0 \in N$ is the initial node,*
- *$I : N \to \mathcal{B}(\mathcal{C})$ assigns an invariant to each node,*
- *$E \subseteq N \times \mathcal{B}(\mathcal{C}) \times \mathcal{A} \times 2^{\mathcal{C}} \times N$ is the set of edges.*

When $\langle l, g, a, r, l'\rangle \in E$, we write $l \xrightarrow{g,a,r} l'$. □

We shall present the operational semantics for timed automata in terms of a process algebraic language in which each term denotes an automaton.

2.2 The Language

We assume a given set of process variables, ranged over by X, Y, ... The language for timed automata over $\mathcal{C}$ can be given by the following BNF grammar:

$$\begin{array}{rcl} s & ::= & \{\psi\}t \;\mid\; \mathbf{fix}X\,s \\ t & ::= & \mathbf{0} \;\mid\; \phi{\to}t \;\mid\; t+t \;\mid\; a(\mathbf{x}).s \;\mid\; a(\mathbf{x}).X \end{array}$$

where ϕ, ψ ranges over $\mathcal{B}(\mathcal{C})$.

$\mathbf{0}$ is the inactive process which can do nothing, except for allowing time to pass. $\phi{\to}t$, read "if ϕ then t", is the usual (one-armed) conditional construct. $a(\mathbf{x}).t$ is the action prefixing operation with clock resetting. $+$ is the nondeterministic choice. The $\{\psi\}t$ construct introduces an invariant. Note that invariants can only occur at places which correspond to locations in timed automata. For instance, strings having the forms $\phi{\to}\{\psi\}t$, $\{\phi\}t+\{\psi\}u$ or $\{\phi\}\{\psi\}t$ are *not* terms of the language, while $\{\psi\}(t+u)$ and $\phi{\to}a(\mathbf{x}).\{\psi\}t$ are allowed. This syntax is specially designed for timed automata, in the sense that there is an exact correspondence between terms of the language and timed automata, as shall be manifested by the symbolic operational semantics below. Later when discussing completeness in Section 5 we shall relax the syntax a little, by including terms of the form $\phi{\to}\{\psi\}t$ which do not denote timed automata by themselves but are useful vehicles for reasoning about equivalence between terms representing timed automata.

A recursion $\mathbf{fix}Xt$ binds X in t. This is the only binding operator in the language. It induces the notions of bound and free process variables as usual. The set of free process variables of t is denoted by $FV(t)$. Terms not containing free process variables

$$\text{DELAY}\quad \frac{}{t\rho \xrightarrow{d} t(\rho+d)}\ \rho+d \models Inv(t) \qquad \text{CHOICE}\quad \frac{t\rho \xrightarrow{a} t'\rho'}{(t+u)\rho \xrightarrow{a} t'\rho'}$$

$$\text{ACTION}\quad \frac{}{(a(\mathbf{x}).t)\rho \xrightarrow{a} t\rho\{\mathbf{x}:=0\}} \qquad \text{GUARD}\quad \frac{t\rho \xrightarrow{a} t'\rho'}{(\phi\to t)\rho \xrightarrow{a} t'\rho'}\ \rho \models \phi$$

$$\text{REC}\quad \frac{(t[\mathbf{fix}Xt/X])\rho \xrightarrow{a} t'\rho'}{(\mathbf{fix}Xt)\rho \xrightarrow{a} t'\rho'} \qquad \text{INV}\quad \frac{t\rho \xrightarrow{a} t'\rho'}{(\{\phi\}t)\rho \xrightarrow{a} t'\rho'}\ \rho \models \phi$$

Figure 2: Standard Transitional Semantics

are *closed*. A recursion $\mathbf{fix}Xt$ is *guarded* if every occurrence of X in t is within the scope of an action prefixing. A term is guarded if every recursion in it is guarded. Note that according to the BNF grammar, in this language all recursions are guarded.

The set of clock variables used in a term t is denoted $\mathcal{C}(t)$.

A *clock valuation* is a function from $\mathcal{C}$ to $\mathbf{R}^{\geq 0}$ (non-negative real numbers), and we use ρ to range over clock valuations. The notations $\rho\{\mathbf{x}:=0\}$ and $\rho+d$ are defined by

$$\rho\{\mathbf{x}:=0\}(y) = \begin{cases} 0 & \text{if } y \in \mathbf{x} \\ \rho(y) & \text{otherwise} \end{cases}$$
$$(\rho+d)(x) = \rho(x)+d \quad \text{for all } x$$

To give a transitional semantics to our language, we first assign each term t an invariant constraint $Inv(t)$ by defining

$$Inv(t) = \begin{cases} \phi & \text{if } t \equiv \{\phi\}t' \\ Inv(s) & \text{if } t \equiv \mathbf{fix}Xs \\ \mathsf{tt} & \text{otherwise} \end{cases}$$

We shall require that all invariants are downward closed:

$$\text{For all } d \in \mathbf{R}^{\geq 0},\ \rho+d \models \phi \text{ implies } \rho \models \phi$$

Given a clock valuation $\rho : \mathcal{C} \to \mathbf{R}^{\geq 0}$, a term can be interpreted according to the rules in Figure 2, where the symmetric rule for + has been omitted. We call $t\rho$ a *process*, where t is a term and ρ a valuation; we use $p,\ q, \ldots$ to range over the set of processes. The transitional semantics uses two types of transition relations: action transition $\xrightarrow{a}$ and delay transition $\xrightarrow{d}$ where d is a non-negative real number. We shall write μ for either an action or a delay. Note that a process can have only a finite number of action transitions, but it may have infinitely many delay transitions. It is the later that makes timed processes infinite branching (and infinite states).

$$\textsc{Action}\quad \frac{}{a(\mathbf{x}).t \xrightarrow{\mathtt{tt},a,\mathbf{x}} t} \qquad \textsc{Choice}\quad \frac{t \xrightarrow{b,a,\mathbf{x}} t'}{t+u \xrightarrow{b,a,\mathbf{x}} t'}$$

$$\textsc{Guard}\quad \frac{t \xrightarrow{\psi,a,\mathbf{x}} t'}{\phi{\rightarrow}t \xrightarrow{\phi\wedge\psi,a,\mathbf{x}} t'} \qquad \textsc{Rec}\quad \frac{t[\mathbf{fix}Xt/X] \xrightarrow{b,a,\mathbf{x}} t'}{\mathbf{fix}Xt \xrightarrow{b,a,\mathbf{x}} t'}$$

$$\textsc{Inv}\quad \frac{t \xrightarrow{\psi,a,\mathbf{x}} t'}{\{\phi\}t \xrightarrow{\psi,a,\mathbf{x}} t'}$$

Figure 3: Symbolic Transitional Semantics

Definition 2 A symmetric relation R over processes is a timed bisimulation if $(p,q) \in R$ implies

whenever $p \xrightarrow{\mu} p'$ then $q \xrightarrow{\mu} q'$ for some q' with $(p',q') \in R$.

We write $p \sim q$ if $(p,q) \in R$ for some timed bisimulation R. □

Lemma 3 *If $t\rho \sim u\rho$ for all $\rho \models \phi$ then $\phi \models Inv(t) \Leftrightarrow Inv(u)$.*

Proof: Let $\rho \models \phi$. Then $t\rho \sim u\rho$. Suppose $\rho \models Inv(t)$. We have $t\rho \xrightarrow{0} t\rho$. This (0-delay) transition must be matched by $u\rho \xrightarrow{0} u\rho$, which implies $\rho \models Inv(u)$. Hence $\phi \models Inv(t) \Rightarrow Inv(u)$. By symmetry we conclude $\phi \models Inv(t) \Leftrightarrow Inv(u)$. □

The symbolic transitional semantics of this language is listed in Figure 3. Again the symmetric rule for + has been omitted. Note that invariants are simply forgotten in the symbolic transitional semantics. This reflects our intention that symbolic transitions correspond to edges in timed automata, while invariants reside in nodes. Note also that there is no rule to deduce "delay" transitions in the symbolic semantics. As noted above, delay transitions are the source of infinity in the semantics of timed automata. The purpose of the symbolic transitional semantics, and the symbolic timed bisimulation built on it, is to avoid such infinity. Thus "delays" do not appear explicitly in the symbolic semantics. Instead they will be implicitly encoded in the notion of "upward-closedness" used to define symbolic bisimulation in the next section.

According to the symbolic semantics, each closed term of the language gives rise to a timed automaton. On the other hand, it is not difficult to see that every timed automaton can be generated from a closed term in the language. In the sequel we will use the phrases "timed automata" and "terms" interchangeably. The two versions of transitional semantics can be related as follows (note again only action transitions are related):

Lemma 4 *1. If $t \xrightarrow{\phi,a,\mathbf{x}} t'$ then $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$ for any $\rho \models \phi \wedge Inv(t)$.*

2. *If* $t\rho \xrightarrow{a} t'\rho'$ *then there exist* ϕ, $\mathbf{x}$ *such that* $\rho \models \phi \wedge Inv(t)$, $\rho' = \rho\{\mathbf{x} := 0\}$ *and* $t \xrightarrow{\phi,a,\mathbf{x}} t'$.

Proof: Both are proved by transition induction.

1. Assuming $t \xrightarrow{\phi,a,\mathbf{x}} t'$ and $\rho \models \phi \wedge Inv(t)$, we show $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$.

- $\phi \equiv \mathsf{tt}$ and $t \equiv a(\mathbf{x}).t' \xrightarrow{\mathsf{tt},a,\mathbf{x}} t'$. Then $(a(\mathbf{x}).t')\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$ by ACTION and $\rho \models \phi$.

- $\phi \equiv \phi' \wedge \psi$ and $t \equiv \phi' \rightarrow t'' \xrightarrow{\phi' \wedge \psi,a,\mathbf{x}} t'$ is derived from $t'' \xrightarrow{\psi,a,\mathbf{x}} t'$. Then $Inv(t) = \mathsf{tt}$. Since $\rho \models \psi \wedge Inv(t)$, by induction we get $t''\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$. Since $\rho \models \phi'$, $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$ by GUARD.

- $Inv(t) = \phi'$ and $t \equiv \{\phi'\}t'' \xrightarrow{\psi,a,\mathbf{x}} t'$ is derived from $t'' \xrightarrow{\psi,a,\mathbf{x}} t'$. Then $Inv(t'') = \mathsf{tt}$, and $\rho \models \psi \wedge Inv(t'')$. By induction, we get $t''\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$. Since $\rho \models \phi'$, $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$ by INV.

- The other cases are similar.

2. Assuming $t\rho \xrightarrow{a} t'\rho'$, we show $t \xrightarrow{\phi,a,\mathbf{x}} t'$ for some ϕ, $\mathbf{x}$ such that $\rho \models \phi \wedge Inv(t)$ and $\rho' = \rho\{\mathbf{x} := 0\}$.

- $t \equiv a(\mathbf{x}).t'$ and $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$. Then $Inv(t) = \mathsf{tt}$. By ACTION we have $t \xrightarrow{\mathsf{tt},a,\mathbf{x}} t'$ and $\rho \models \mathsf{tt} \wedge Inv(t)$.

- $t \equiv \psi \rightarrow t''$ and $t\rho \xrightarrow{a} t'\rho'$ is derived from $\rho \models \psi$ and $t''\rho \xrightarrow{a} t'\rho'$. Then $Inv(t) = \mathsf{tt}$. By induction we get $t'' \xrightarrow{\phi,a,\mathbf{x}} t'$ for some ϕ, $\mathbf{x}$ such that $\rho \models \phi \wedge Inv(t'')$ and $\rho' = \rho\{\mathbf{x} := 0\}$. By GUARD, $t \xrightarrow{\phi \wedge \psi,a,\mathbf{x}} t'$ and $\rho \models \phi \wedge \psi \wedge Inv(t)$.

- $t \equiv \{\phi'\}t''$ and $t\rho \xrightarrow{a} t'\rho'$ is derived from $\rho \models \phi'$ and $t''\rho \xrightarrow{a} t'\rho'$. Then $Inv(t) = \phi'$ and $Inv(t'') = \mathsf{tt}$. By induction we get $t'' \xrightarrow{\phi,a,\mathbf{x}} t'$ for some ϕ, $\mathbf{x}$ such that $\rho \models \phi$ and $\rho' = \rho\{\mathbf{x} := 0\}$. By INV, $t \xrightarrow{\phi,a,\mathbf{x}} t'$. Also $\rho \models \phi \wedge Inv(t)$.

- The other cases are similar.

□

3 Symbolic Timed Bisimulation

In this section we shall define a symbolic version of timed bisimulation. To simplify the presentation we fix two timed automata. To avoid clock variables of one automaton being reset by the other, we assume the sets of clocks of the two timed automata under

consideration are disjoint, and write C for the union of the two clock sets [1]. Let N be the largest natural number occurring in the constraints of the two automata. An atomic constraint over C with ceiling N has one of the three forms: $x \bowtie m$ or $x - y \bowtie n$ where $x, y \in C, \bowtie \in \{\leq, <, \geq, >\}$ and $m, n \leq N$ are natural numbers.

In the following, "atomic constraint" always means "atomic constraint over C with ceiling N". Note that given two timed automata there are only finite number of such atomic constraints. We shall use c to range over atomic constraints.

A constraint, or *zone*, is a boolean combination of atomic constraints.[2] A constraint ϕ is consistent if there is some ρ such that $\rho \models \phi$. Let ϕ and ψ be two constraints. We write $\phi \models \psi$ to mean $\rho \models \phi$ implies $\rho \models \psi$ for any ρ. Note that the relation $\models$ is decidable.

A *region constraint* [AD94], or *region* for short, over n clock variables $x_1, \ldots, x_n$ is a consistent constraint containing the following atomic conjuncts:

- For each $i \in \{1, \ldots, n\}$ either $x_i = m_i$ or $m_i < x_i < m_i + 1$ or $x_i > N$;
- For each pair of $i, j \in \{1, \ldots, n\}$, $i \neq j$, such that both x_i and x_j are not greater than N, either $x_i - \lfloor x_i \rfloor = x_j - \lfloor x_j \rfloor$ or $x_i - \lfloor x_i \rfloor < x_j - \lfloor x_j \rfloor$ or $x_j - \lfloor x_j \rfloor < x_i - \lfloor x_i \rfloor$.

where $\lfloor x_i \rfloor$ is the integral part of x_i, namely the m_i related to x_i in the first clause.

Given a finite set of clock variables C and a ceiling N, the set of region constraints over C is finite and is denoted $\mathcal{RC}_N^C$. In the sequel, we will omit the sub- and superscripts when they can be supplied by the context.

Fact 5 *Let ϕ be a region constraint. If $\rho \models \phi$ and $\rho' \models \phi$ then*

- *For all $i \in \{1, \ldots, n\}$, if $\rho(x_i) \leq N$ then $\lfloor \rho(x_i) \rfloor = \lfloor \rho'(x_i) \rfloor$.*
- *For any $i, j \in \{1, \ldots, n\}$, $i \neq j$,*
  - *$\{\rho(x_i)\} = \{\rho(x_j)\}$ iff $\{\rho'(x_i)\} = \{\rho'(x_j)\}$, and*
  - *$\{\rho(x_i)\} < \{\rho(x_j)\}$ iff $\{\rho'(x_i)\} < \{\rho'(x_j)\}$.*

where $\{x\}$ is the fractional part of x.

That is, two valuations satisfying the same region constraint must agree on their integral parts as well as on the ordering of their fractional parts.

Lemma 6 *Suppose that ϕ is a region constraint and ψ a zone. Then either $\phi \Rightarrow \psi$ or $\phi \Rightarrow \neg\psi$.*

[1] This does not impose any restriction on our results, because we can always rename clock variables of an automaton without affecting its behaviour.

[2] In this paper a zone is an arbitrary constraint and may not be a convex.

Proof: We first transform ψ into disjunctive normal form: $\psi = \bigvee_i \bigwedge_j e_{ij}$ where each e_{ij} is an atomic constraint. Now $\psi \wedge \phi = \bigvee_i \bigwedge_j (e_{ij} \wedge \phi)$. It is easy to see, by examining the possible forms of e_{ij}, that each $e_{ij} \wedge \phi$ is either equal to ϕ or false. Hence $\psi \wedge \phi$ is either equal to ϕ or false. In the former case we have $\phi \Rightarrow \psi$, and in the later case we get $\phi \Rightarrow \neg\psi$. □

According to this lemma, a region is either entirely contained in a zone, or is completely outside a zone. In other words, regions are the finest polyhedra that can be described by our constraint language.

Lemma 6 has a useful corollary. Let $\mathcal{RC}(\psi)$ be the set of regions contained in the zone ψ, *i.e.* $\mathcal{RC}(\psi) = \{\, \phi \in \mathcal{RC} \mid \phi \Rightarrow \psi \,\}$.

Corollary 7 $\bigvee \mathcal{RC}(\psi) \Leftrightarrow \psi$.

Proof: By definition, $\bigvee \mathcal{RC}(\psi) \Rightarrow \psi$. To show $\psi \Rightarrow \bigvee \mathcal{RC}(\psi)$, let $\rho \models \psi$. There must be some $\phi \in \mathcal{RC}$ such that $\rho \models \phi$. Since $\phi \not\Rightarrow \neg\psi$, by Lemma 6 we have $\phi \Rightarrow \psi$. Hence $\phi \in \mathcal{RC}(\psi)$. Therefore $\rho \models \bigvee \mathcal{RC}(\psi)$. □

The notion of a region constraint enjoys an important property: processes in the same region behave uniformly with respect to timed bisimulation ([Cer92]):

Fact 8 *Let t, u be two terms with disjoint sets of clock variables and ϕ a region constraint over the union of the two clock sets. Suppose that both ρ and ρ' satisfy ϕ. Then $t\rho \sim u\rho$ iff $t\rho' \sim u\rho'$.*

A *canonical constraint* is a disjunction of regions. ψ is a *canonical form* of ϕ if ψ is a canonical constraint and $\psi \Leftrightarrow \phi$. According to Corollary 7, $\bigvee \mathcal{RC}(\phi)$ is a canonical form of ϕ. Given a constraint we can first transform it into disjunctive normal form, then decompose each disjunct into a disjoint set of regions. Both steps can be effectively implemented.

We will need two (postfixing) operators to deal with resetting. The first one is $\downarrow_{\mathbf{x}}$ where $\mathbf{x} \subseteq C$. We first define it on regions, then generalise it to zones. With abuse of notation, we will write $c \in \phi$ to mean c is a conjunct of ϕ.

For a region ϕ,

$$\begin{aligned} \phi\downarrow_{\mathbf{x}} = \phi\downarrow'_{\mathbf{x}} \quad & \wedge \ \bigwedge\{\, x_i = 0 \mid x_i \in \mathbf{x} \,\} \wedge \bigwedge\{\, x_i = x_j \mid x_i, x_j \in \mathbf{x} \,\} \\ & \wedge \ \bigwedge\{\, x_i = x_j - m \mid x_i \in \mathbf{x}, x_j \notin \mathbf{x}, x_j = m \in \phi \,\} \\ & \wedge \ \bigwedge\{\, x_i < x_j - m \mid x_i \in \mathbf{x}, x_j \notin \mathbf{x}, x_j > m \in \phi \,\} \end{aligned}$$

and $\downarrow'_{\mathbf{x}}$ is defined by

$$\begin{array}{ll} \mathsf{tt}\downarrow'_{\mathbf{x}} = \mathsf{tt} & \\ (c \wedge \phi)\downarrow'_{\mathbf{x}} = \phi\downarrow'_{\mathbf{x}} & \text{if } \mathbf{x} \cap fv(c) \neq \emptyset \\ (c \wedge \phi)\downarrow'_{\mathbf{x}} = c \wedge \phi\downarrow'_{\mathbf{x}} & \text{if } \mathbf{x} \cap fv(c) = \emptyset \end{array}$$

where $fv(c)$ is the set of clock variables appearing in (atomic constraint) c.

For a canonical constraint $\bigvee_i \phi_i$ with each ϕ_i a region, $(\bigvee_i \phi_i)\downarrow_{\mathbf{x}} = \bigvee_i (\phi_i\downarrow_{\mathbf{x}})$. For an arbitrary constraint ϕ, $\phi\downarrow_{\mathbf{x}}$ is understood as the result of applying $\downarrow_{\mathbf{x}}$ to the canonical form of ϕ.

Lemma 9 *1. $\rho \models \phi$ implies $\rho\{\mathbf{x} := 0\} \models \phi{\downarrow_{\mathbf{x}}}$.*

2. If ϕ is a region constraint then so is $\phi{\downarrow_{\mathbf{x}}}$.

Proof:

1. Let $\bigvee_i \phi_i$ be the canonical form of ϕ. Since $\rho \models \phi$, $\rho \models \phi_i$ for some i. Now ϕ_i is a region constraint, so $\rho\{\mathbf{x} := 0\} \models \phi_i{\downarrow_{\mathbf{x}}}$ follows immediately from the definition of $\downarrow_{\mathbf{x}}$.

2. Immediately from the definition of $\downarrow_{\mathbf{x}}$.

□

The second operator $\Uparrow$ is defined similarly. We first define it on regions:

$$\phi\Uparrow = \phi\Uparrow' \wedge \bigwedge_{i \leq j} e_{ij}(\phi)$$

where $\Uparrow'$ is defined by

$$\begin{array}{l}
(x < m \wedge \phi)\Uparrow' = \phi\Uparrow' \\
(x = m \wedge \phi)\Uparrow' = m \leq x \wedge \phi\Uparrow' \\
(x_i - m_i < x_j - m_j \wedge \phi)\Uparrow' = x_i - m_i < x_j - m_j \wedge \\
\qquad\qquad\qquad\qquad\qquad\qquad x_j - m_j < x_i - m_i + 1 \wedge \phi\Uparrow' \\
(c \wedge \phi)\Uparrow' = c \wedge \phi\Uparrow' \quad \text{for other atomic constraint } c \\
\mathsf{tt}\Uparrow' = \mathsf{tt}
\end{array}$$

and

$$e_{ij}(\phi) = \begin{cases} x_i - m_i = x_j - m_j & \text{if } x_i = m_i, x_j = m_j \in \phi \\ \mathsf{tt} & \text{otherwise} \end{cases}$$

For an arbitrary constraint ϕ, $\phi\Uparrow$ is understood as the result of applying $\Uparrow$ to each disjunct of the canonical form of ϕ. ϕ is $\Uparrow$-*closed* if and only if $\phi\Uparrow = \phi$.

Lemma 10 *1. $\rho \models \phi$ implies $\rho \models \phi\Uparrow$.*

2. $\phi\Uparrow$ is $\Uparrow$-closed.

3. If ϕ is $\Uparrow$-closed then $\rho \models \phi$ implies $\rho + d \models \phi$ for all $d \in \mathbf{R}^{\geq 0}$.

Proof:

1. Immediately from the definition of $\Uparrow$.

2. It is sufficient to consider the case when ϕ is a region constraint. We check if each conjunct introduced by the $\Uparrow$ operator is preserved by a further application of it. The only interesting case is the third clause in the definition of $\Uparrow'$. Direct calculation gives:

$$\begin{array}{ll} & (x_i - m_i < x_j - m_j \wedge x_j - m_j < x_i - m_i + 1)\Uparrow' \\ = & (x_i - m_i < x_j - m_j \wedge x_j - m_j < x_i - m_i + 1)\wedge \\ & (x_j - m_j < x_i - m_i + 1 \wedge x_i - m_i + 1 < x_j - m_j + 1) \\ = & x_i - m_i < x_j - m_j \wedge x_j - m_j < x_i - m_i + 1 \end{array}$$

3. Let $\bigvee_i \phi_i$ be the canonical form of ϕ. We have $\phi\Uparrow = \bigvee_i \phi_i\Uparrow = \phi$. Since $\rho \models \phi$, $\rho \models \phi_i\Uparrow$ for some i. It is straightforward to check that $\rho + d \models \psi$ for each conjunct ψ of $\phi_i\Uparrow$.

□

Symbolic bisimulation will be defined as a family of binary relations indexed by clock constraints. Following [Cer92] we use constraints over the union of the (disjoint) clock sets of two timed automata as indices. The reason for this is as follows: the definition of timed bisimulation requires two processes to match action transitions as well as delay transitions, which amounts to requiring them to match action transitions *while their clocks progress at the same rate.* In the definition of symbolic bisimulation indexing constraints are subject to the $\Uparrow$ operation that introduces equalities between clock variables (the e_{ij} component in the above definition), which guarantees the "same rate" requirement when such constraints are over the union of the two clock sets.

Given a constraint ϕ, a finite set of constraints Φ is called a *ϕ-partition* if $\bigvee \Phi = \phi$. In particular $\emptyset$ is a ff-partition. A ϕ-partition Φ is called *finer* than another such partition Ψ if Φ can be obtained from Ψ by decomposing some of its elements. By Corollary 7, $\mathcal{RC}(\phi)$ is a ϕ-partition, and is the finest such partition. In particular, if ϕ is a region constraint then $\{\phi\}$ is the only partition of ϕ.

Definition 11 A constraint indexed family of symmetric relations over terms $\mathbf{S} = \{ S^\phi \mid \phi \text{ is } \Uparrow\text{−closed} \}$ is a symbolic timed bisimulation if $(t, u) \in S^\phi$ implies

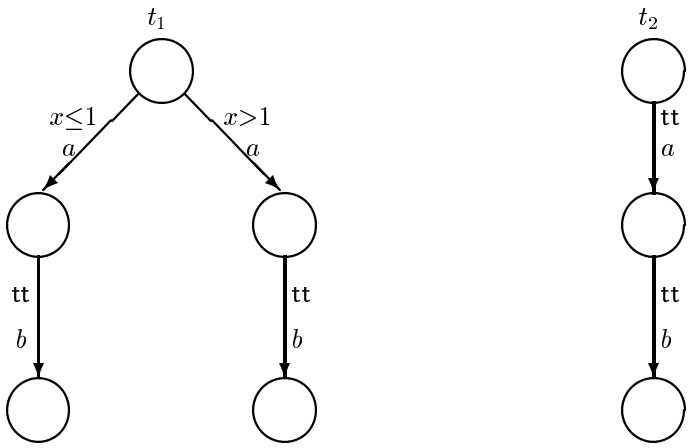
1. $\phi \models Inv(t) \Leftrightarrow Inv(u)$ and

2. whenever $t \xrightarrow{\psi, a, \mathbf{x}} t'$ then there is a $Inv(t) \wedge \phi \wedge \psi$-partition Φ such that for each $\phi' \in \Phi$ there is $u \xrightarrow{\psi', a, \mathbf{y}} u'$ for some ψ', $\mathbf{y}$ and u' such that $\phi' \Rightarrow \psi'$ and $(t', u') \in S^{\phi'\downarrow_{\mathbf{xy}}\Uparrow}$.

We write $t \sim^\phi u$ if $(t, u) \in S^\phi \in \mathbf{S}$ for some symbolic bisimulation $\mathbf{S}$. □

Note that there is no clause for delay transitions in this definition, because delays are encoded in the $\Uparrow$-closedness property of the indexing constraints and in the $\phi \models Inv(t) \Leftrightarrow Inv(u)$ requirement.

The use of a partition when matching a symbolic transition is essential. Without it we would not be able to characterise timed bisimulation using symbolic transitions. For example, consider the two timed automata t_1 and t_2 below (we have omitted the empty resets). They are apparently timed bisimilar. But the symbolic transition $t_2 \stackrel{\mathtt{tt},a,\{\}}{\longrightarrow}$ can not be entirely matched by either of the two symbolic transitions from t_1. We must use a partition, say $\{x \leq 1,\ x > 1\}$: t_1 can match the symbolic transition from t_2 using its left branch over the constraint $x \leq 1$, and the right branch over $x > 1$.



Symbolic timed bisimulation captures $\sim$ in the following sense:

Theorem 12 *For $\Uparrow$−closed ϕ, $t \sim^{\phi} u$ iff $t\rho \sim u\rho$ for any $\rho \models \phi$.*

Proof: ($\Longrightarrow$) Assume $(t, u) \in S^{\phi} \in \mathbf{S}$ for some symbolic bisimulation $\mathbf{S}$. Define

$$R = \{\ (t\rho, u\rho) \mid \text{for some } \phi,\ \rho \models \phi \text{ and } (t, u) \in S^{\phi} \in \mathbf{S}\ \}$$

We show R is a timed bisimulation. Suppose $(t\rho, u\rho) \in R$, i.e. there is some ϕ such that $\rho \models \phi$ and $(t, u) \in S^{\phi}$. By the first clause in Definition 11, we have $\rho \models Inv(t)$ if and only if $\rho \models Inv(u)$.

- $t\rho \stackrel{a}{\longrightarrow} t'\rho'$. By Lemma 4 there are ψ, $\mathbf{x}$ such that $\rho \models \psi \wedge Inv(t)$, $\rho' = \rho\{\mathbf{x} := 0\}$ and $t \stackrel{\psi,a,\mathbf{x}}{\longrightarrow} t'$. So there is a $Inv(t) \wedge \phi \wedge \psi$-partition Φ with the properties specified in Definition 11. Since $\rho \models \phi \wedge \psi$, $\rho \models \phi'$ for some $\phi' \in \Phi$. Let $u \stackrel{\psi',a,\mathbf{y}}{\longrightarrow} u'$ be the symbolic transition associated with this ϕ', as guaranteed by Definition 11. Then $\phi' \Rightarrow \psi'$ and $(t', u') \in S^{\phi'\downarrow_{\mathbf{xy}}\Uparrow}$. Since $\rho \models \psi' \wedge Inv(u)$, $u\rho \stackrel{a}{\longrightarrow} u'\rho\{\mathbf{y} := 0\}$. By Lemma 9, $\rho\{\mathbf{xy} := 0\} \models \phi'\downarrow_{\mathbf{xy}}$. By Lemma 10, $\rho\{\mathbf{xy} := 0\} \models \phi'\downarrow_{\mathbf{xy}}\Uparrow$. Therefore $(t'\rho\{\mathbf{xy} := 0\}, u'\rho\{\mathbf{xy} := 0\}) \in R$. Since $t'\rho\{\mathbf{xy} := 0\} \equiv t'\rho\{\mathbf{x} := 0\}$ and $u'\rho\{\mathbf{xy} := 0\} \equiv u'\rho\{\mathbf{y} := 0\}$, this is the same as $(t'\rho\{\mathbf{x} := 0\}, u'\rho\{\mathbf{y} := 0\}) \in R$.

- $t\rho \xrightarrow{d} t(\rho + d)$. Since ϕ is $\Uparrow$-closed, $\rho + d \models \phi$. Then $\rho + d \models Inv(u)$ and hence $u\rho \xrightarrow{d} u(\rho + d)$. Therefore $(t(\rho + d), u(\rho + d)) \in R$.

($\Longleftarrow$) Assume that for any $\rho \models \phi_0$, $t\rho \sim u\rho$, *i.e.* $(t\rho, u\rho) \in R$ for some timed bisimulation R, we show $t \sim^{\phi_0} u$ as follows. For each $\Uparrow-$closed ϕ, define

$$S^{\phi} = \{\, (t, u) \mid \forall \phi' \in \mathcal{RC}(\phi),\ (t\rho, u\rho) \in R \text{ for any } \rho \models \phi' \,\}$$

and let $\mathbf{S} = \{\, S^{\phi} \mid \phi \text{ is } \Uparrow-\text{closed} \,\}$. Then $(t, u) \in S^{\phi_0}$. By Fact 8 $\mathbf{S}$ is well-defined. We show $\mathbf{S}$ is a symbolic bisimulation. Suppose $(t, u) \in S^{\phi}$. Then for all $\phi' \in \mathcal{RC}(\phi)$, for any $\rho \models \phi'$, $(t\rho, u\rho) \in R$. By Lemma 3, for all $\phi' \in \mathcal{RC}(\phi)$, $\phi' \models Inv(t) \Leftrightarrow Inv(u)$. Therefore $\phi \models Inv(t) \Leftrightarrow Inv(u)$. Now let $t \xrightarrow{\psi, a, \mathbf{x}} t'$. Define $\Phi' = \{\, \phi' \mid \phi' \in \mathcal{RC}(\phi) \text{ and } \phi' \Rightarrow \psi \,\}$. Then Φ' is a $\phi \wedge \psi$-partition. For each $\phi' \in \Phi'$, there exists ρ s.t. $\rho \models \phi'$ with $(t\rho, u\rho) \in R$. By the definition of Φ', $\rho \models \psi$. By Lemma 4, $t\rho \xrightarrow{a} t'\rho\{\mathbf{x} := 0\}$. Since $(t\rho, u\rho) \in R$, $u\rho \xrightarrow{a} u'\rho'$ for some u' and ρ' with $(t'\rho\{\mathbf{x} := 0\}, u'\rho') \in R$. By Lemma 4 again, $u \xrightarrow{\psi', a', \mathbf{y}} u'$ for some ψ' and $\mathbf{y}$ with $\rho \models \psi'$ and $\rho' = \rho\{\mathbf{y} := 0\}$. Hence $(t\rho\{\mathbf{x} := 0\}, u\rho\{\mathbf{y} := 0\}) \in R$, which is the same as $(t\rho\{\mathbf{xy} := 0\}, u\rho\{\mathbf{xy} := 0\}) \in R$. Since $\rho \models \phi'$, by Lemma 9 we have $\rho\{\mathbf{xy} := 0\} \models \phi'{\downarrow_{\mathbf{xy}}}$. Since ϕ' is a region constraint, so is $\phi'{\downarrow_{\mathbf{xy}}}$ which is the only element of $\mathcal{RC}(\phi'{\downarrow_{\mathbf{xy}}})$. Therefore $(t', u') \in S^{\phi'{\downarrow_{\mathbf{xy}}}\Uparrow}$. □

4 The Proof System

The proposed proof system consists of a set of inference rules in Figure 4 and a set of equational axioms in Figure 5. The judgments of the inference system are *conditional equations* of the form

$$\phi \rhd t = u$$

where ϕ is a constraint and t, u are terms. Its intended meaning is "$t \sim^{\phi} u$", or "$t\rho \sim u\rho$ for any $\rho \models \phi \wedge Inv(t) \wedge Inv(u)$". $\mathsf{tt} \rhd t = u$ will be abbreviated as $t = u$.

The axioms are the standard monoid laws for bisimulation in process algebras. More interesting are the inference rules. For each construct in the language there is a corresponding introduction rule. CHOICE expresses the fact that timed bisimulation is preserved by $+$. The rule GUARD does a case analysis on condition. The rule INV deals with invariants. It also performs a case analysis and looks very similar to GUARD. However, there is a crucial difference: When the guard ψ is false $\psi{\rightarrow}t$ behaves like $\mathbf{0}$, the process which is inactive but can allow time to pass; On the other hand, when the invariant ψ is false $\{\psi\}t$ behaves like $\{\mathsf{ff}\}\mathbf{0}$, the process usually referred to as *time-stop*, which is not only inactive but also "still", can not even let time elapse. ACTION is the introduction rule for action prefixing (with clock resetting). The THINNING rule allows to introduce/remove redundant clocks. REC is the usual rule for folding/unfolding recursions, while UFI says $\mathbf{fix}Xu$ is the unique solution of the equation $X = u$ (recall that all recursions in our language are guarded). Finally the two rules PARTITION

EQUIV $\dfrac{}{t = t} \quad \dfrac{\phi \rhd t = u}{\phi \rhd u = t} \quad \dfrac{\phi \rhd t = u \quad \phi \rhd u = v}{\phi \rhd t = v}$

AXIOM $\dfrac{}{t = u}$ $t = u$ is an axiom instance

CHOICE $\dfrac{\phi \rhd t = t'}{\phi \rhd t + u = t' + u}$

GUARD $\dfrac{\phi \wedge \psi \rhd t = u \quad \phi \wedge \neg\psi \rhd \mathbf{0} = u}{\phi \rhd \psi \to t = u}$

INV $\dfrac{\phi \wedge \psi \rhd t = u \quad \phi \wedge \neg\psi \rhd \{\mathsf{ff}\}\mathbf{0} = u}{\phi \rhd \{\psi\}t = u}$

ACTION $\dfrac{\phi{\downarrow_{\mathbf{x}}}\Uparrow \rhd t = u}{\phi \rhd a(\mathbf{x}).t = a(\mathbf{x}).u}$

THINNING $\dfrac{}{a(\mathbf{xy}).t = a(\mathbf{x}).t} \quad \mathbf{y} \cap \mathcal{C}(t) = \emptyset$

REC $\dfrac{}{\mathbf{fix}Xt = t[\mathbf{fix}Xt/X]}$

UFI $\dfrac{t = u[t/X]}{t = \mathbf{fix}Xu}$

PARTITION $\dfrac{\phi_1 \rhd t = u \quad \phi_2 \rhd t = u}{\phi \rhd t = u} \quad \phi \models \phi_1 \vee \phi_2$

ABSURD $\dfrac{}{\mathsf{ff} \rhd t = u}$

Figure 4: The Inference Rules

and ABSURD do not handle any specific constructs in the language. They are so-called "structural rules" used to "glue" pieces of derivation together.

Taking $\phi_1 = \phi_2$ PARTITION specialises to a useful rule

$$\text{CONSEQUENCE} \quad \frac{\phi_1 \rhd t = u}{\phi \rhd t = u} \quad \phi \models \phi_1$$

Let us write $\vdash \phi \rhd t = u$ to mean $\phi \rhd t = u$ can be derived from this proof system. Some useful properties of the proof system are summarised in the following propo-

S1 $t + \mathbf{0} = t$ S2 $t + t = t$

S3 $t + u = u + t$ S4 $(t + u) + v = t + (u + v)$

Figure 5: The Equational Axioms

sition:

Proposition 13 *1.* $\vdash \phi{\rightarrow}(\psi{\rightarrow}t) = \phi \wedge \psi{\rightarrow}t$

2. $\vdash t = t + \phi{\rightarrow}t$

3. If $\phi \models \psi$ *then* $\vdash \phi \rhd t = \psi{\rightarrow}t$

4. $\vdash \phi \wedge \psi \rhd t = u$ *implies* $\vdash \phi \rhd \psi{\rightarrow}t = \psi{\rightarrow}u$

5. $\vdash \phi{\rightarrow}(t + u) = \phi{\rightarrow}t + \phi{\rightarrow}u$

6. $\vdash \phi{\rightarrow}t + \psi{\rightarrow}t = \phi \vee \psi{\rightarrow}t$

Proof: We only give proofs for *1* and *4*, leaving the others to the readers.
We first prove a lemma:

$$\text{If } \phi \wedge \psi = \mathsf{ff} \text{ then } \vdash \phi \rhd \psi{\rightarrow}t = \mathbf{0} \tag{1}$$

By GUARD we need to show

$$\phi \wedge \psi \rhd t = \mathbf{0} \text{ and } \phi \wedge \neg\psi \rhd \mathbf{0} = \mathbf{0}$$

The first follows from the hypothesis and ABSURD, while the second from EQUIV.
1. An application of GUARD gives

$$\phi \rhd \psi{\rightarrow}t = \phi \wedge \psi{\rightarrow}t \tag{2}$$

and

$$\neg\phi \rhd \mathbf{0} = \phi \wedge \psi{\rightarrow}t \tag{3}$$

(3) is an instance of (1). To show (2), we apply GUARD again and obtain two subgoals:

$$\phi \wedge \psi \rhd t = \phi \wedge \psi{\rightarrow}t \text{ and } \phi \wedge \neg\psi \rhd \mathbf{0} = \phi \wedge \psi{\rightarrow}t$$

The second subgoal can be settled by (1). Apply symmetry (the middle rule in EQUIV) followed by GUARD, the first subgoal is reduced to

$$(\phi \wedge \psi) \wedge (\phi \wedge \psi) \rhd t = t \text{ and } (\phi \wedge \psi) \wedge \neg(\phi \wedge \psi) \rhd \mathbf{0} = t$$

which can be settled by EQUIV (plus CONSEQUENCE) and ABSURD, respectively.

4. By GUARD, $\vdash \phi \rhd \psi{\rightarrow}t = \psi{\rightarrow}u$ can be reduced to

$$\phi \wedge \psi \rhd t = \phi{\rightarrow}u \quad \text{and} \quad \phi \wedge \neg\psi \rhd \mathbf{0} = \phi{\rightarrow}u$$

The second subgoal is an instance of (1). For the first one we apply GUARD again obtaining

$$(\phi \wedge \psi) \wedge \psi \rhd t = u \quad \text{and} \quad (\phi \wedge \psi) \wedge \neg\psi \rhd t = \mathbf{0}$$

Now the first subgoal follows from the hypothesis and the second from ABSURD. □

The following lemma shows how to "push" a condition through an action prefix:

Lemma 14 $\vdash \phi \rhd a(\mathbf{x}).t = a(\mathbf{x}).\phi{\downarrow_{\mathbf{x}}}{\Uparrow}{\rightarrow}t$.

Proof: By ACTION this can be reduced to

$$\vdash \phi{\downarrow_{\mathbf{x}}}{\Uparrow} \rhd t = \phi{\downarrow_{\mathbf{x}}}{\Uparrow}{\rightarrow}t$$

which follows from Proposition 13.*3*. □

The rule PARTITION has a more general form:

Proposition 15 *Suppose* Ψ *is a* ϕ*-partition and* $\vdash \psi \rhd t = u$ *for each* $\psi \in \Psi$*, then* $\vdash \phi \rhd t = u$.

Proof: By induction on the size of Ψ. The base case when Ψ contains only one element is trivial. For the induction step, assume the statement of the proposition holds for ϕ-partitions of size k and let $\Psi = \{\, \psi_i \mid 1 \leq i \leq k{+}1 \,\}$. Set $\Psi' = \{\, \neg\psi_{k+1}{\wedge}\psi_i \mid 1 \leq i \leq k \,\}$. Since $\vdash \psi_i \rhd t = u$, by CONSEQUENCE we have $\vdash \neg\psi_{k+1} \wedge \psi_i \rhd t = u$. Therefore by the induction hypothesis,

$$\vdash \bigvee \Psi' \rhd t = u$$

From this and the assumption $\vdash \psi_{k+1} \rhd t = u$, by PARTITION we obtain

$$\vdash \psi_{k+1} \vee \bigvee \Psi' \rhd t = u$$

Since $\psi_{k+1} \vee \bigvee \Psi' = \psi_{k+1} \vee (\neg\psi_{k+1} \wedge \bigvee_{1 \leq i \leq k} \psi_i) = \bigvee_{1 \leq i \leq k+1} \psi_i = \bigvee \Psi = \phi$, this completes the induction. □

In the rest of this section we discuss the soundness of the proof system. First we show that the rule UFI is sound with respect to $\sim$. Following [Mil89] we use the technique of *bisimulation up to*.

Definition 16 A symmetric relation R is a *timed bisimulation up to* $\sim$ if $(p, q) \in R$ implies

- whenever $p \xrightarrow{d} p'$ then $q \xrightarrow{d} q'$ for some q' and $(p', q') \in R$.

- whenever $p \xrightarrow{a} p'$ then $q \xrightarrow{a} q'$ for some q' and $(p', q') \in \sim R \sim$.

□

Note that the derivatives of delay transitions are required to be in the same relation, while those of action transitions are allowed to be related modulo $\sim$.

Lemma 17 *If R is a timed bisimulation up to $\sim$ then $R \subseteq \sim$.*

Proof: Let $(p, q) \in R$ and $p \xrightarrow{\mu} p'$. We need to show that there is some q' such that $q \xrightarrow{\mu} q'$ and $(p', q') \in R$. The case when μ is an action is settled in the same way as in the proof of Proposition 6, Section 4.3, [Mil89]. The case when μ is a delay follows directly from Definition 16. □

Lemma 18 *If $v[t/X] \xrightarrow{a} t'$, then t' has the form $v'[t/X]$, and moreover, for any u, $v[u/X] \xrightarrow{a} v'[u/X]$.*

This lemma concerns only action transitions and its proof is the same as that of Lemma 13, Section 4.5, [Mil89].

Proposition 19 *Suppose $fv(v) \subseteq \{X\}$. If $t\rho \sim v[t/X]\rho$ and $u\rho \sim v[u/X]\rho$ then $t\rho \sim u\rho$.*

Proof: We show the relation

$$R = \{\, (v[t/X]\rho, v[u/X]\rho) \mid fv(v) \subseteq \{X\} \,\}$$

is a timed bisimulation up to $\sim$. Assume $(v[t/X]\rho, v[u/X]\rho) \in R$ and consider the following two cases, according to the types of transitions:

- $v[t/X]\rho \xrightarrow{d} v[t/X](\rho + d)$. Then also $v[u/X]\rho) \xrightarrow{d} v[u/X](\rho + d)$ and $(v[t/X](\rho + d), v[u/X](\rho + d)) \in R$.
- $v[t/X]\rho \xrightarrow{a} t'\rho'$. In this case we can find a matching transition $v[u/X]\rho \xrightarrow{a} u'\rho'$ such that $(t'\rho', u'\rho') \in \sim R \sim$, as in the proof of Proposition 14, Section 4.5, [Mil89].

By symmetry we conclude R is a timed bisimulation up to $\sim$. □

Soundness of the proof system is stated below:

Theorem 20 *If $\vdash \phi \rhd t = u$ and ϕ is $\Uparrow$-closed then $t\rho \sim u\rho$ for any $\rho \models \phi$.*

The standard approach to the soundness proof is by induction on the length of derivations, and perform a case analysis on the last rule/axiom used. However, this does not quite work here. The reason is that the definition of timed bisimulation requires two processes to simulate each other after any time delays. To reflect this in the proof system, we apply the $\Uparrow$ operator, after $\downarrow_{\mathbf{x}}$ for clock resetting, in the premise of the ACTION rule. But not all the inference rules preserve the $\Uparrow$-closedness property. An example is GUARD. In order to derive $\phi \rhd \psi \rightarrow t = u$, we need to establish $\phi \wedge \psi \rhd t = u$ and $\phi \wedge \neg\psi \rhd \mathbf{0} = u$. Even if ϕ is $\Uparrow$-closed, $\phi \wedge \psi$ may not be so.

To overcome this difficulty, we introduce the notion of "timed bisimulation up to a time bound", formulated as follows

Definition 21 Two processes p and q are timed bisimular up to $d_0 \in \mathbf{R}^{\geq 0}$, written $p \sim^{d_0} q$, if for any d such that $0 \leq d \leq d_0$

- whenever $p \xrightarrow{d} p'$ then $q \xrightarrow{d} q'$ for some q' and $p' \dot{\sim} q'$,
- whenever $q \xrightarrow{d} q'$ then $p \xrightarrow{d} p'$ for some p' and $p' \dot{\sim} q'$.

where $p \dot{\sim} q$ is defined thus

- whenever $p \xrightarrow{a} p'$ then $q \xrightarrow{a} q'$ for some q' and $p' \sim q'$,
- whenever $q \xrightarrow{a} q'$ then $p \xrightarrow{a} p'$ for some p' and $p' \sim q'$.

□

The difference between timed bisimulation up to d and the standard notion of timed bisimulation only concerns initial delay transitions: in timed bisimulation up to d two processes are required to match only those initial delay transitions with duration no greater that d. Note that $\dot{\sim}$ is the same as $\sim^0$, and $\sim^{d_1} \subseteq \sim^{d_2}$ for $d_1 \geq d_2$.

Lemma 22 *1. If $p \sim^{d_0} q$ for any $d_0 \in \mathbf{R}^{\geq 0}$ then $p \sim q$.*

2. Let ρ_i and d_i, $0 \leq i \leq n$, be such that $\rho_{i+1} = \rho_i + d_i$, $0 \leq i < n$. If $t\rho_i \sim^{d_i} u\rho_i$ for all i such that $0 \leq i \leq n$, then $t\rho_0 \sim^{d} u\rho_0$ where $d = d_0 + \cdots + d_n$.

Proof:

1. Let *Nat* be the set of natural numbers and $\sim^{\infty} = \bigcap_{i \in Nat} \sim^i$. We show $\sim^{\infty} = \sim$. By definition $\sim \subseteq \sim^i$ for any i, hence $\sim \subseteq \sim^{\infty}$. To show $\sim^{\infty} \subseteq \sim$, *i.e.* $\sim^{\infty}$ is a timed bisimulation, assume that $(t\rho, u\rho') \in \sim^{\infty}$, *i.e.* $(t\rho, u\rho') \in \sim^i$ for any $i \in Nat$, and consider two kinds of transitions from $t\rho$.

- $t\rho \xrightarrow{d} t(\rho+d)$. Take an i_0 with $i_0 > d$. Since $(t\rho, u\rho') \in \sim^{i_0}$, $u\rho' \xrightarrow{d} u(\rho'+d)$ and $(t(\rho+d), u(\rho'+d)) \in \dot{\sim}$. To see $(t(\rho+d), u(\rho'+d)) \in \sim^{\infty}$, let $t(\rho+d) \xrightarrow{d'} t(\rho+d+d')$. Then $t\rho \xrightarrow{d+d'} t(\rho+d+d')$. Take an i_1 with $i_1 > d'$. Since $(t\rho, u\rho') \in \sim^{i_1}$, $u\rho' \xrightarrow{d+d'} u(\rho'+d+d')$ (hence $u(\rho'+d) \xrightarrow{d'} u(\rho'+d+d')$) and $(t(\rho+d+d'), u(\rho'+d+d')) \in \dot{\sim}$.
- $t\rho \xrightarrow{a} p'$. Since $(t\rho, u\rho') \in \dot{\sim}$, $u\rho' \xrightarrow{a} q'$ for some q' and $(p', q') \in \sim$. Since $\sim \subseteq \sim^{\infty}$, $(p', q') \in \sim^{\infty}$.

By symmetry, we conclude $\sim^{\infty} \subseteq \sim$.

Now if $p \sim^{d_0} q$ for any $d_0 \in \mathbf{R}^{\geq 0}$, then $p \sim^{\infty} q$, hence $p \sim q$.

2. By induction on n. The base case when $n = 0$ is trivial. Now suppose the result holds for $n-1$ and consider the case for n. By the induction hypothesis, $t\rho_0 \sim^{d'} u\rho_0$ where $d' = d_0 + \cdots + d_{n-1}$. Let $t\rho_0 \xrightarrow{d''} t(\rho_0 + d'')$ with $d'' \leq d = d' + d_n$. If $d'' \leq d'$, then, since $t\rho_0 \sim^{d'} u\rho_0$, we have $u\rho_0 \xrightarrow{d''} u(\rho_0 + d'')$ and $t(\rho_0 + d'') \dot{\sim} u(\rho_0 + d'')$. Otherwise $d' < d'' \leq d_n$. Consider the transition $t\rho_0 \xrightarrow{d'} t(\rho_0 + d')$. We have $u\rho_0 \xrightarrow{d'} u(\rho_0 + d')$. Since $\rho_0 + d' = \rho_n$, $0 < d'' - d' \leq d_n$, and $t\rho_n \sim^{d_n} u\rho_n$, $t(\rho_0 + d') = t\rho_n \xrightarrow{d''-d'} t(\rho_n + d'' - d') = t(\rho_0 + d'')$. Hence $u(\rho_0 + d') = u\rho_n \xrightarrow{d''-d'} u(\rho_n + d'' - d') = u(\rho_0 + d'')$ (that is $u\rho_0 \xrightarrow{d''} u(\rho_0 + d'')$) and $t(\rho_0 + d'') \dot{\sim} u(\rho_0 + d'')$. By symmetry, $t\rho_0 \sim^{d} u\rho_0$.

□

Now the following proposition, of which Theorem 20 is a special case when ϕ is $\Uparrow$-closed, can be proved by standard induction on the length of derivations :

Proposition 23 *If $\vdash \phi \rhd t = u$ then $t\rho \sim^{d_0} u\rho$ for any ρ and d_0 such that $\rho + d \models \phi$ for all $0 \leq d \leq d_0$.*

Proof: By induction on the length of inference. The base case when the length is 0 is straightforward. For the induction step we do case analysis on the last rule applied.

- ACTION. Assume ρ, d_0 are such that $\rho + d \models \phi$ for any $0 \leq d \leq d_0$. To show $(a(\mathbf{x}).t)\rho \sim^{d_0} (a(\mathbf{x}).u)\rho$, let $(a(\mathbf{x}).t)\rho \xrightarrow{d} (a(\mathbf{x}).t)(\rho + d)$ with $0 \leq d \leq d_0$. This can be matched by $(a(\mathbf{x}).u)\rho \xrightarrow{d} (a(\mathbf{x}).u)(\rho + d)$. To see $(a(\mathbf{x}).t)(\rho + d) \dot{\sim} (a(\mathbf{x}).u)(\rho + d)$, observe that the only possible action transition from both sides is an a-transition:

$$(a(\mathbf{x}).t)(\rho + d) \xrightarrow{a} t(\rho + d)\{\mathbf{x} := 0\}$$
$$(a(\mathbf{x}).u)(\rho + d) \xrightarrow{a} u(\rho + d)\{\mathbf{x} := 0\}$$

Write ρ' for $(\rho + d)\{\mathbf{x} := 0\}$. For any d_0' and any $0 \leq d' \leq d_0'$, $\rho' + d' \models \phi\downarrow_{\mathbf{x}}\Uparrow$. By induction, $t\rho' \sim^{d_0'} u\rho'$. Therefore $t\rho' \sim u\rho'$ by Lemma 22. This establishes $(a(\mathbf{x}).t)(\rho + d) \dot{\sim} (a(\mathbf{x}).u)(\rho + d)$. Hence $(a(\mathbf{x}).t)\rho \sim^{d_0} (a(\mathbf{x}).u)\rho$.

- GUARD. Assume ρ, d_0 are such that $\rho + d \models \phi$ for any $0 \leq d \leq d_0$. The line $[\rho,\ \rho + d_0]$ is divided by regions into finitely many segments $[\rho_0, \rho_1)$, $[\rho_1, \rho_1]$, (ρ_1, ρ_2), $[\rho_2, \rho_2]$, ..., $[\rho_n, \rho_n]$ and $(\rho_n, \rho + d_0]$, where $\rho_0 = \rho$ and $\rho_{i+1} = \rho_i + d'_i$, such that each segment is entirely contained in some region. By Lemma 22 we only need to show $(\psi \rightarrow t)\rho_i \sim^{d'_i} u\rho_i$ for each $1 \leq i \leq n$.

  By Lemma 6, each point $[\rho_i, \rho_i]$ or interval $(\rho_i, \rho_i + d'_i)$ is either entirely contained in zone $\phi \wedge \psi$ or entirely contained in zone $\phi \wedge \neg\psi$. By induction, in the former case we have $(\psi \rightarrow t)\rho_i \sim^{d'_i} u\rho_i$. In the later case we have $\mathbf{0}\rho_i \sim^{d'_i} u\rho_i$. Since $\rho_i + d' \not\models \psi$ for any $0 \leq d' \leq d'_i$ in this case, it follows $\mathbf{0}(\rho_i + d') \sim (\psi \rightarrow t)(\rho_i + d')$. Hence $(\psi \rightarrow t)\rho_i \sim^{d'_i} u\rho_i$. Therefore we have $(\psi \rightarrow t)\rho_i \sim^{d'_i} u\rho_i$ in both cases.

- UFI. See Proposition 19.

- The other cases are similar.

□

5 Completeness

This section is devoted to proving the completeness of the proof system which is stated thus: if $t \sim^{\phi} u$ then $\vdash \phi \rhd t = u$. The structure of the proof follows from that of [Mil84]. The intuition behind the proof is as follows: A timed automaton is presented as a set of standard equations in which the left hand-side of each equation is a formal process variable corresponding to a node of the automaton, while the right hand-side encodes the outgoing edges from the node. We first transform, within the proof system, both t and u into such equation sets (Proposition 24). Using the fact that t and u are timed bisimilar over ϕ as a guide, we can construct a "product" of the two equation sets in such a way that both t and u provably ϕ-satisfy the product equation set (Proposition 25). Finally we demonstrate that a standard set of equations has only one solution, therefore the required equality between t and u can be derived. The unique fixpoint induction is only employed in the last step of the proof, namely Proposition 27.

In carrying out the completeness proof we need to extend the syntax of the language presented at the beginning of Section 2.2, by allowing terms of the form $\phi \rightarrow s$ where s may have the forms $\{\psi\}t$, X or $\mathbf{fix}Xs'$. The operational semantics for $\phi \rightarrow s$ is given by the rule GUARD in Figure 2, exactly the same as those "normal" conditional terms. Since the proof of the soundness theorem for the inference system relies only on the operational semantics of Figure 2 and does not refer to the symbolic operational semantics, the theorem remains valid over the extended language. The new terms, which are not subject to symbolic operational semantics and therefore do not denote timed automata, will only be used in the proofs of Propositions 25 and 27. Thus terms mentioned in the statements of Propositions 24, 25, 27, as well as Theorem 28, are of the non-extended language denoting timed automata.

Let $\mathbf{X} = \{ X_i \mid i \in I \}$ and $\mathbf{W}$ be two disjoint sets of process variables and $\mathbf{x}$ a set of clock variables. Let also u_i, $i \in I$, be terms with free process variables in $\mathbf{X} \cup \mathbf{W}$ and clock variables in $\mathbf{x}$. Then

$$E: \quad \{ X_i = u_i \mid i \in I \}$$

is an equation set with formal process variables $\mathbf{X}$ and free process variables in $\mathbf{W}$. E is closed if $\mathbf{W} = \emptyset$. E is a *standard* equation set if each u_i has the form

$$\{\psi_i\}(\sum_{k \in K_i} \phi_{ik} \rightarrow a_{ik}(\mathbf{x}_{ik}).X_{f(i,k)} + \sum_{k' \in K'_i} \psi_{ik'} \rightarrow W_{f'(i,k')})$$

A term t provably ϕ-satisfies an equation set E if there exist a vector of terms $\{ t_i \mid i \in I \}$ and a vector of conditions $\{ \phi_i \mid i \in I \}$ such that $\phi_1 = \phi$, $\vdash \phi \rhd t_1 = t$, $\phi_i \models Inv(u_i) \Leftrightarrow Inv(t_i)$, and

$$\vdash \phi_i \rhd t_i = u_i[\phi_i \rightarrow t_i / X_i | i \in I]$$

for each $i \in I$. We will simply say "t provably satisfies E" when $\phi_i = \mathsf{tt}$ for all $i \in I$.

Proposition 24 *For any term t with free process variables $\mathbf{W}$ there exists a standard equation set E, with free process variables in $\mathbf{W}$, which is provably satisfied by t. In particular, if t is closed then E is also closed.*

Proof: By induction on the structure of t. The only non-trivial case is recursion when $t \equiv \mathbf{fix} X t'$. By induction there is a standard equation set $E' : \{ X_i = u_i \mid i \in I \}$ with free process variables in $FV(t) \cup \{X\}$ and $t'_i : s$ such that $\vdash t' = t'_1$ and

$$\vdash t'_i = u_i[t'_i / X_i | i \in I]$$

We may assume that X is different from any X_i. Let $v_i = u_i[u_1/X]$ for each i. Note that since X is under an action prefixing in t', it does not occur free in u_1. Hence $v_1 = u_1$. Consider the equation set

$$E: \quad \{ X_i = v_i \mid i \in I \}$$

Set $t_i = t'_i[t/X]$. Then

$$\begin{array}{rcl} \vdash t & = & \mathbf{fix} X t' \\ & = & \mathbf{fix} X t'_1 \\ & \stackrel{REC}{=} & t'_1[\mathbf{fix} X t'_1 / X] \\ & = & t'_1[t/X] \\ & = & t_1 \end{array}$$

and

$$\begin{aligned} \vdash t &= t_1'[t/X] \\ &= u_1[t_i'/X_i | i \in I][t/X] \\ &= u_1[t_i'[t/X]/X_i | i \in I] \\ &= u_1[t_i/X_i | i \in I] \end{aligned}$$

Now

$$\begin{aligned} \vdash t_i &= t_i'[t/X] \\ &= u_i[t_i'/X_i | i \in I][t/X] \\ &= u_i[t, t_i'[t/X]/X, X_i | i \in I] \\ &= u_i[t, t_i/X, X_i | i \in I] \\ &= u_i[u_1[t_i/X_i | i \in I], t_i/X, X_i | i \in I] \\ &= u_i[u_1/X][t_i/X_i | i \in I] \\ &= v_i[t_i/X_i | i \in I] \end{aligned}$$

This shows t satisfies E. □

Proposition 25 *For closed terms t and u, if $t \sim^{\phi} u$ then there exist a ϕ' such that $\phi \Rightarrow \phi'$ and a standard, closed equation set E which is provably ϕ'-satisfied by both t and u.*

Proof: Let the sets of clock variables of $t,\ u$ be $\mathbf{x},\ \mathbf{y}$, respectively, with $\mathbf{x} \cap \mathbf{y} = \emptyset$. Let also E_1 and E_2 be the standard equation sets for t and u, respectively:

$$\begin{aligned} E_1 : &\quad \{\, X_i = \{\phi_i\} \sum_{k \in K_i} \phi_{ik} \rightarrow a_{ik}(\mathbf{x}_{ik}).X_{f(i,k)} \mid i \in I \,\} \\ E_2 : &\quad \{\, Y_j = \{\psi_j\} \sum_{l \in L_j} \psi_{jl} \rightarrow b_{jl}(\mathbf{y}_{jl}).Y_{g(j,l)} \mid j \in J \,\} \end{aligned}$$

So there are $t_i,\ u_j$ such that $\vdash t_1 = t$, $\vdash u_1 = u$, $\models \phi_i \Leftrightarrow Inv(t_i)$, $\models \psi_i \Leftrightarrow Inv(u_j)$, and

$$\begin{aligned} &\vdash t_i = \{\phi_i\} \textstyle\sum_{k \in K_i} \phi_{ik} \rightarrow a_{ik}(\mathbf{x}_{ik}).t_{f(i,k)} \\ &\vdash u_j = \{\psi_j\} \textstyle\sum_{l \in L_j} \psi_{jl} \rightarrow b_{jl}(\mathbf{y}_{jl}).u_{g(j,l)} \end{aligned}$$

Without loss of generality, we may assume $a_{ik} = b_{jl} = a$ for all i, k, j, l.

For each pair of $i,\ j$, let

$$\Phi_{ij} = \{\, \Delta \in \mathcal{RC}(\mathbf{xy}) \mid t_i \sim^{\Delta\Uparrow} u_j \,\}$$

Set $\varphi_{ij} = \bigvee \Phi_{ij}$. By the definition of Φ_{ij}, φ_{ij} is the weakest condition over which t_i and u_j are symbolically bisimilar, that is, $\psi \Rightarrow \varphi_{ij}$ for any ψ such that $t_i \sim^{\psi} u_j$. In

particular, $\phi \Rightarrow \varphi_{11}$. Also for any $\Delta \in \Phi_{ij}$, $\Delta \models Inv(t_i) \Leftrightarrow Inv(u_j)$, hence $\Delta \models \phi_i \Leftrightarrow \psi_j$.

For each $\Delta \in \Phi_{ij}$ let

$$I_{ij}^{\Delta} = \{\, (k,l) \mid \Delta \models \phi_{ik} \wedge \psi_{jl}, t_{f(i,k)} \sim^{\Delta\downarrow_{\mathbf{x}_{ik}\mathbf{y}_{jl}}\Uparrow} u_{g(j,l)} \,\}$$

Define

$$E: \quad Z_{ij} = \{\phi_i\} \sum_{\Delta \in \Phi_{ij}} \Delta \rightarrow \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).Z_{f(i,k)g(j,l)}$$

We claim that E is provably φ_{11}-satisfied by t when each Z_{ij} is instantiated with t_i over φ_{ij}. We need to show, for each i,

$$\vdash \varphi_{ij} \rhd t_i = \{\phi_i\} \sum_{\Delta \in \Phi_{ij}} \Delta \rightarrow (\sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).\varphi_{f(i,k)g(j,l)} \rightarrow t_{f(i,k)})$$

Since the elements of Φ_{ij} are mutually disjoint, by Propositions 15 and 13, it is sufficient to show that, for each $\Delta \in \Phi_{ij}$,

$$\vdash \Delta \rhd t_i = \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).\varphi_{f(i,k)g(j,l)} \rightarrow t_{f(i,k)}$$

By the definition of I_{ij}^{Δ}, we have $t_{f(i,k)} \sim^{\Delta\downarrow_{\mathbf{x}_{ik}\mathbf{y}_{jl}}\Uparrow} u_{g(j,l)}$. Hence, from the definition of Φ_{ij},

$$\Delta\downarrow_{\mathbf{x}_{ik}\mathbf{y}_{jl}}\Uparrow \Rightarrow \varphi_{f(i,k)g(j,l)}$$

Therefore

$$\begin{aligned}
\vdash \Delta \quad & \rhd \quad \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).\varphi_{f(i,k)g(j,l)} \rightarrow t_{f(i,k)} \\
& \overset{\textit{Lemma 14}}{=} \quad \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).\Delta\downarrow_{\mathbf{x}_{ik}\mathbf{y}_{jl}}\Uparrow \rightarrow \varphi_{f(i,k)g(j,l)} \rightarrow t_{f(i,k)} \\
& \overset{\textit{Prop. 13}}{=} \quad \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).\Delta\downarrow_{\mathbf{x}_{ik}\mathbf{y}_{jl}}\Uparrow \rightarrow t_{f(i,k)} \\
& \overset{\textit{Lemma 14}}{=} \quad \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}\mathbf{y}_{jl}).t_{f(i,k)} \\
& \overset{\text{THINNING}}{=} \quad \{\phi_i\} \sum_{(k,l)\in I_{ij}^{\Delta}} a(\mathbf{x}_{ik}).t_{f(i,k)} \\
& \overset{\text{S1-S4}}{=} \quad \{\phi_i\} \sum_{k\in K_i} a(\mathbf{x}_{ik}).t_{f(i,k)} \\
& \overset{\textit{Prop. 13.3}}{=} \quad \{\phi_i\} \sum_{k\in K_i} \phi_{ik} \rightarrow a(\mathbf{x}_{ik}).t_{f(i,k)} \\
& = \quad t_i
\end{aligned}$$

Symmetrically we can show that E is provably ϕ_{11}-satisfied by u when Z_{ij} is instantiated with u_j over ϕ_{ij}. □

Now we come to the last step in the completeness proof which requires the UFI rule. This rule, as presented in Figure 4, is unconditional. However, a conditional version is not difficult to derive.

Lemma 26 *From* $\vdash \phi \rhd t = u[\phi{\to}t/X]$ *infer* $\vdash \phi \rhd t = \mathbf{fix}X(\phi{\to}u)$.

Proof: Assume $\vdash \phi \rhd t = u[\phi{\to}t/X]$. By Proposition 13.*4*,

$$\vdash \phi{\to}t = \phi{\to}u[\phi{\to}t/X]$$

By UFI,

$$\vdash \phi{\to}t = \mathbf{fix}X(\phi{\to}u)$$

Hence

$$\begin{array}{lcl} \vdash \phi{\to}t & \stackrel{\text{REC}}{=} & (\phi{\to}u)[\mathbf{fix}X(\phi{\to}u)/X] \\ & = & \phi{\to}u[\mathbf{fix}X(\phi{\to}u)/X] \\ & \stackrel{Prop.\ 13}{=} & \phi{\to}\phi{\to}u[\mathbf{fix}X(\phi{\to}u)/X] \\ & = & \phi{\to}(\phi{\to}u)[\mathbf{fix}X(\phi{\to}u)/X] \\ & \stackrel{\text{REC}}{=} & \phi{\to}\mathbf{fix}X(\phi{\to}u) \end{array}$$

Therefore, by By Proposition 13.*3*,

$$\vdash \phi \rhd t = \mathbf{fix}X(\phi{\to}u).$$

□

Proposition 27 *If both* t *and* u *provably* ϕ*-satisfy an equation set* E *then* $\vdash \phi \rhd t = u$.

Proof: By induction on the size of E. For the base case when E contains only one equation $X_1 = v_1$, we have $\vdash \phi \rhd t = v_1[\phi{\to}t/X_1]$. By Lemma 26, $\vdash \phi \rhd t = \mathbf{fix}X_1\phi{\to}v_1$. Similarly $\vdash \phi \rhd u = \mathbf{fix}X_1\phi{\to}v_1$. Hence $\vdash \phi \rhd t = u$.

Assume the result for m and let E contain $m+1$ equations:

$$X_i = v_i \quad 1 \le i \le m+1$$

Since t provably ϕ-satisfies E, there are t_i and ϕ_i, $1 \le i \le m+1$, such that $\vdash t_1 = t$, $\phi_1 = \phi$, and

$$\vdash \phi_i \rhd t_i = v_i[\phi_i{\to}t_i/X_i | 1 \le i \le m+1]$$

for each $1 \le i \le m+1$. In particular

$$\begin{array}{lcl} \vdash \phi_{m+1} \rhd t_{m+1} & = & v_{m+1}[\phi_i{\to}t_i/X_i | 1 \le i \le m+1] \\ & = & (v_{m+1}[\phi_i{\to}t_i/X_i | 1 \le i \le m])[\phi_{m+1}{\to}t_{m+1}/X_{m+1}] \end{array}$$

By Lemma 26,

$$\vdash \phi_{m+1} \rhd t_{m+1} = \mathbf{fix}X_{m+1}(\phi_{m+1}{\rightarrow}v_{m+1}[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m])$$

Writing w_{m+1} for $\mathbf{fix}X_{m+1}(\phi_{m+1}{\rightarrow}v_{m+1})$, We have

$$\vdash \phi_{m+1} \rhd t_{m+1} = w_{m+1}[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m]$$

By Proposition 13,

$$\vdash \phi_{m+1}{\rightarrow}t_{m+1} = \phi_{m+1}{\rightarrow}w_{m+1}[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m]$$

Therefore

$$\begin{array}{ll} \vdash & \phi_i \rhd t_i = v_i[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m+1] \\ = & v_i[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m][\phi_{m+1}{\rightarrow}t_{m+1}/X_{m+1}] \\ = & v_i[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m][\phi_{m+1}{\rightarrow}w_{m+1}[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m]/X_{m+1}] \\ = & (v_i[\phi_{m+1}{\rightarrow}w_{m+1}/X_{m+1}])[\phi_i{\rightarrow}t_i/X_i|1 \leq i \leq m] \end{array}$$

This shows t provably ϕ-satisfies the equation set

$$E': \quad X_i = v_i[\phi_{m+1}{\rightarrow}w_{m+1}/X_{m+1}] \quad 1 \leq i \leq m$$

Symmetrically we can show that u provably ϕ-satisfies E'. By induction we conclude $\vdash \phi \rhd t = u$. □

Putting together Propositions 25 and 27 we obtain the main theorem of this section:

Theorem 28 *For closed terms t and u, if $t \sim^{\phi} u$ then $\vdash \phi \rhd t = u$.*

Proof: By Proposition 25, there is a standard equation set E which are ϕ'-satisfied by both t and u for some ϕ' such that $\phi' \Rightarrow \phi$. By Proposition 27, $\vdash \phi' \rhd t = u$. Finally, by CONSEQUENCE, $\vdash \phi \rhd t = u$. □

6 Conclusion and Related Work

We have presented an axiomatisation, in the form of an inference system, of timed bisimulation for timed automata, and proved its soundness and completeness. To this end a CCS-like language is introduced. The correspondence between terms of this language and timed automata are established via a symbolic style operational semantics. We have shown that by extending pure equational reasoning with a set of inference rules dealing with specific language constructs needed for timed automata, the standard monoid laws for bisimulation are sufficient for characterizing bisimulation in the timed world. This result agrees with the previous work on proof systems for value-passing

processes [HL96] and for π-calculus [Lin95], providing a further evidence that the four monoid laws capture the essence of bisimulation.

To the best of our knowledge, this is the first complete axiomatisation for the full set of timed automata. It fills a gap in the theory of timed automata, and demonstrates that bisimulation equivalence of timed automata are as mathematically tractable as those of standard process algebras.

A classic result in automata theory is complete axiomatisation of language equivalence for finite state automata in terms of regular expressions [Sal66]. For timed automata it has been shown that the problem of deciding timed language equivalence is Π_1^1 hard [AD94]. However, the more discriminating relation, timed bisimulation, is decidable [Cer92]. The most recent development in algebraic characterisations for timed automata are presented in [ACM97, BP99]. As the main result, they established that each timed automaton is equivalent to an algebraic expression out of the standard operators in formal languages, such as union, intersection, concatenation and variants of Kleene's star operator, in the sense that the automaton recognize the same timed language as denoted by the expression. However, the issue of axiomatisation was not considered there. In [DAB96] a set of equational axioms was proposed for timed automata, but no completeness result was reported. [BS98] presents an algebraic framework for real-time systems which is similar to timed automata where "invariants" are replaced by "deadlines" (to express "urgency"), together with some equational laws. Apart from these, we are not aware of any other published work on axiomatising timed automata. On the other hand, most timed extensions of process algebras came with equational axiomatisations. For instance, [Bor96] developed a symbolic theory for a timed process algebra which includes a proof system for its finite fragment, while [AJ94] used unique fixpoint induction to achieve a complete axiomatisation for the regular subset of the timed-CCS proposed in [Wan91].

Acknowledgments The authors would like to thank the anonymous referees for their constructive comments. In particular the suggestion from a referee has strengthened the result of Theorem 3.6.

References

[ACM97] E. Asarin, P. Caspi and O. Maler. A Kleene theorem for timed automata. In proceedings of LICS'97. pp.160-171. 1997.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235. 1994.

[AJ94] L. Aceto and A. Jeffrey. A complete axiomatization of timed bisimulation for a class of timed regular behaviours. *Theoretical Computer Science*, 152(2):251–268. 1995.

[Bor96] M. Boreale. Symbolic Bisimulation for Timed Processes. In Proceedings of *AMAST'96*, LNCS 1101, pp.321-335. Springer–Verlag. 1996.

[BP99] P. Bouyer and A. Petit. Decomposition and Composition of Timed Automata. In *ICALP'99*, LNCS 1644, pp. 210-219. Springer–Verlag. 1999.

[BS98] S. Bornot and J. Sifakis. An Algebraic Framework for Urgency. Information and Computation, 163:172-202, 2000.

[Cer92] K. Čeräns. Decidability of Bisimulation Equivalences for Parallel Timer Processes. In *CAV'92*, LNCS 663, pp.302-315. Springer–Verlag. 1992.

[DAB96] P.R. D'Argenio and Ed Brinksma. A Calculus for Timed Automata (Extended Abstract). In *FTRTFTS'96*, LNCS 1135, pp.110-129. Springer–Verlag. 1996.

[HL95] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–389. 1995.

[HL96] M. Hennessy and H. Lin. Proof systems for message-passing process algebras. *Formal Aspects of Computing*, 8:408–427. 1996.

[Lin95] H. Lin. Unique Fixpoint Induction for the π-Calculus In Proceedings of CONCUR'95, LNCS 962, pp.88–102. Springer–Verlag. 1995.

[LW00] H. Lin and Y. Wang. A proof system for timed automata. In Proceedings of Fossacs'2000, LNCS 1784, pp.208–222. Springer–Verlag. March 2000.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *J. Computer and System Science*, 28:439–466. 1984.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall. 1989.

[Sal66] A. Salomaa. Two Complete Axiom Systems for the Algebra of Regular Events. *Journal of the ACM*, 13:158-169. 1966.

[Wan91] Wang Yi. *A Calculus of Real Time Systems*. Ph.D. thesis, Chalmers University. 1991.

[WPD94] Wang Yi, Paul Pettersson, and Mats Daniels. Automatic Verification of Real-Time Communicating Systems By Constraint-Solving. In *Proc. of the 7*th *International Conference on Formal Description Techniques*. pp.223-238. 1994.