A calculus for timed automata*

Pedro R. D'Argenio and Ed Brinksma

Dept. of Computer Science. University of Twente.

P.O.Box 217. 7500 AE Enschede. The Netherlands.

{*dargenio,brinksma*}*@cs.utwente.nl*

June, 1996

Abstract

A language for representing timed automata is introduced. Its semantics is defined in terms of timed automata. This language is complete in the sense that any timed automaton can be represented by a term in the language. We also define a direct operational semantics for the language in terms of (timed) transition systems. This is proven to be equivalent (or, more precisely, timed bisimilar) to the interpretation in terms of timed automata.

In addition, a set of axioms is given that is shown to be sound for timed bisimulation. Finally, we introduce several features like hiding operator, the parallel composition and derived time operations like wait, time-out and urgency. We conclude with an example and show that we can eliminate non-reachable states using algebraic techniques.

1991 Mathematics Subject Classification: 68Q45, 68Q55, 68Q60.

1991 CR Categories: D.3.1, F.3.1, F.3.2, F.4.3.

Keywords: process algebra, real time, timed automata, timed transition system.

Note: An extended abstract of this report was accepted to be published in the *Proceedings of the Fourth International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems, FTRTFT'96*, Uppsala, Sweden, September 1996.

*Supported by the NWO/SION project 612-33-006.

Contents

1 Introduction

A real-time system is a system whose behaviour is constrained by requirements on the time in which events can occur. Sometimes, systems are implemented as timed systems in the sense that they fulfil certain timing conditions to give them an acceptable performance. Other systems depend on timing conditions in a more essential way, viz. because their functional correctness depends upon certain critical timing conditions being fulfilled. Therefore, it becomes interesting to study the formal verification of such systems.

In the last years, several formal techniques have been developed to specify and verify real-time systems. For instance, many well-known process algebras have been extended with features to manipulate time [Dav92, Yi90, MT90, NS94, BB91, Klu91, Klu93, BB95, BL92, LL94]. But the apparently most successful approaches are timed and hybrid automata [AD94, NSY92, HNSY94, ACH$^+$95]. The formal relation between these two models has been studied in some cases [NSY92, NSY93, Fok94, DOY94]. Languages that fully represent timed automata have also been studied [LV94, YPD94].

In this paper, we introduce a process algebra to describe timed automata. Since the syntax of timed automata becomes unwieldy to specify realistic real-time systems, the process algebra introduced here proposes a higher-level language that is interpreted in terms of timed automata. More specifically, we choose a slight variation of the so called timed safety automata [HNSY94]. Basically, the language extends Milner's CCS [Mil89] restricted to prefixing, inaction and summation, with some features to manipulate clocks, namely, clock resetting, invariants and guards. We prove that any timed automaton can be described by a term in the language together with guarded recursion.

Also, we introduce a direct operational semantics for the language. Thus, a (timed) transition system is associated to each process. We prove that this way of giving semantics is equivalent (*timed bisimilar*) to the interpretation of the associated timed automaton.

In order to facilitate the construction of complex system we include the usual process operations, hiding and parallel composition, and several common operations on time, such as time-out, waiting and urgency.

The first goal of our paper is to introduce a powerful language to represent timed automata. Our second goal is to introduce an equational theory for the language that allows us to manipulate timed automata in order to eliminate redundant information. This is an interesting point, since, to our knowledge, timed automata have not yet been studied from an algebraic point of view. The axiomatisation is sound for timed bisimulation and allows to find a normal form. Moreover, the additional operators like hiding and parallel composition can be eliminated, thus obtaining equivalent expressions defined just in terms of the basic language.

As an example we study the railroad crossing controller of [AD94]. In this example, we illustrate that we can eliminate redundant states, clocks, and conditions. In particular, non-reachable states are eliminated.

The rest of this paper is structured as follows. Section 2 reviews the models of timed transition systems and timed automata. In Section 3, we introduce the language and we study its relation with timed automata. The operational semantics is introduced in Section 4 and the relation with the timed automata model is stated. Section 5 introduces the axiomatisation for the basic language, and the extension with new operators is studied in Section 6. The example is presented in Section 7. Extensionality with respect to CCS, related work, and conclusions are discussed in Section 8.

Acknowledgements. This work profited from discussions with Jan F. Groote, Rom Langerak, Jan Springintveld, Jan Tretmans, Frits Vaandrager and Sergio Yovine. In particular, Sergio Yovine pointed out the related work [YPD94] and Jan Springintveld pointed out the connection with [HKWT95]. Reference [AH94] was pointed out by one of the referees of FTRTFT'96.

2 Models for Timed Systems

2.1 Time, Clocks and Constraints

We adopt the set $\mathbb{R}^{\geq 0}$ of non-negative reals as *time domain*. A *clock* is a variable x ranging over a time domain $\mathbb{R}^{\geq 0}$. Let $\mathcal{C}$ denote a set of clocks. The set $\Phi(\mathcal{C})$ of *clock constraints* over $\mathcal{C}$ is defined inductively by:

$$\phi ::= \quad d \leq d' \;\mid\; x \leq d \;\mid\; d \leq x \;\mid\; x - y \leq d \;\mid\; d \leq x - y \;\mid\; (\phi \wedge \phi) \;\mid\; (\neg\phi)$$

where $d, d' \in \mathbb{R}^{\geq 0}$ and $x, y \in \mathcal{C}$ with $x \neq y$. The abbreviations $\mathbf{tt}$, $\mathbf{ff}$, $x = d$, $x > d$, $x \in [d, d')$, $x - y < d$, $\phi \vee \phi'$, $\phi \Rightarrow \phi'$, etc. are defined as usual. Let $var(\phi)$ denote the set of clocks occurring in ϕ. A clock constraint is closed if no clocks occur in it. We denote the set of closed clock constraints by Φ^c. We could also adopt a richer set of constraint (see Section 8).

An *assignment* is a function $v : \mathcal{C} \to \mathcal{C} \cup \mathbb{R}^{\geq 0}$. Let $\mathcal{V}$ denote the set of assignments. v is lifted to clocks constraints by the obvious induction over the structure of ϕ. We also lift an assignment v to $\wp(\mathcal{C})$ as usual: $v(C) = \{v(x) | x \in C\}$. Let $f : C \to C'$, with $C \subseteq \mathcal{C}$ and $C' \subseteq \mathcal{C} \cup \mathbb{R}^{\geq 0}$. We define $v[f]$ as follows:

$$v[f](x) \stackrel{\text{def}}{=} \begin{cases} f(x) & \text{if } x \in C \\ v(x) & \text{if } x \notin C \end{cases}$$

We write $[x \mapsto w]$ for $f : \{x\} \to \{w\}$ with $f(x) = w$ and $[C \mapsto w]$ for $f : C \to \{w\}$ with $f(x) = w$ for all $x \in C$. Let $d \in \mathbb{R}^{\geq 0}$. Define $v + d$ as follows:

$$(v + d)(x) \stackrel{\text{def}}{=} v(x) + d$$

Let $v \circ v'$ be the *composition* of assignments defined for all $x \in \mathcal{C}$, $v \circ v'(x) \stackrel{\text{def}}{=} v(v'(x))$. Notice that $(v \circ v') + d = (v + d) \circ v'$ Let ι be the *identity assignment*. An assignment v is a *renaming* if for all $x \in \mathcal{C}$, $v(x) \in \mathcal{C}$. An assignment v is a *valuation* if for all $x \in \mathcal{C}$,

$v(x) \in \mathbb{R}^{\geq 0}$. Let $\mathcal{V}^c \subseteq \mathcal{V}$ be the set of all valuations. Notice that for any valuation v and for any clock constraint ϕ, $v(\phi)$ is a closed clock constraint. For the subset of closed clock constraints, we define the satisfaction predicate $\models \subseteq \Phi^c$ as usual:

$$\frac{}{\models d \leq d + d'} \qquad \frac{\models \phi \quad \models \phi'}{\models (\phi \wedge \phi')} \qquad \frac{\not\models \phi}{\models (\neg\phi)}$$

where $d, d' \in \mathbb{R}^{\geq 0}$. We generalise $\models$ to all clock constraints ($\models \subseteq \Phi(\mathcal{C})$). Let $\phi \in \Phi(\mathcal{C})$ then

$$\models \phi \stackrel{\text{def}}{\iff} \forall v \in \mathcal{V}^c : \models v(\phi)$$

We define the set $\overline{\Phi}(\mathcal{C}) \subseteq \Phi(\mathcal{C})$ of *past-closed constraints* as follows:

$$\phi \in \overline{\Phi}(\mathcal{C}) \stackrel{\text{def}}{\iff} \forall v \in \mathcal{V}^c, d \in \mathbb{R}^{\geq 0}.\ \models (v + d)(\phi) \implies \models v(\phi)$$

Notice that this kind of constraints are such that if they hold at time d, they hold at all $d' < d$.

2.2 Timed Transition Systems

A timed transition system is a labelled transition system that includes information about the time. We adopt the model of actions with *time stamps*.

Definition 2.1 (Timed transition systems)
Let $\mathbf{A}$ be a set of *actions*. A *timed transition system* (TTS) is a structure $L = (S, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_0, \longrightarrow, \mathcal{U})$ where

- S is a set of *states*, with the *initial state* $s_0 \in S$;
- A is a set of *labels*;
- $\longrightarrow \subseteq S \times (\mathbf{A} \times \mathbb{R}^{\geq 0}) \times S$ is the *transition relation*; and
- $\mathcal{U} \subseteq \mathbb{R}^{\geq 0} \times S$ is the *until predicate*.

We use the following notation: $a(d)$ iff $(a, d) \in \mathbf{A} \times \mathbb{R}^{\geq 0}$, $s \xrightarrow{a(d)} s'$ iff $\langle s, a(d), s' \rangle \in \longrightarrow$, $\mathcal{U}_d(s)$ iff $\langle d, s \rangle \in \mathcal{U}$, $s \xrightarrow{a(d)}$ iff $\exists s' \in S.\ s \xrightarrow{a(d)} s'$ and $s \not\xrightarrow{a(d)}$ iff $\neg(s \xrightarrow{a(d)})$.

In addition, L should satisfies the following axioms:

Until $\forall d, d' \in \mathbb{R}^{\geq 0}.\ \mathcal{U}_d(s) \wedge d' < d \implies \mathcal{U}_{d'}(s)$;

Delay $\forall d \in \mathbb{R}^{\geq 0}.\ s \xrightarrow{a(d)} \implies \mathcal{U}_d(s)$. □

The intended meaning of a transition $s \xrightarrow{a(d)} s'$ is that a system which is in state s can change to be in state s' by performing an action a at time d. Intuitively, $\mathcal{U}_d(s)$ together with axiom **Until**, means that a system can idle in a state s at least d units of times. Axiom **Delay** state that every time that an action may occur in a state s at time d, the system must be idling at that time.

Predicate $\mathcal{U}$ was introduced in [Klu93]. Here, we formalised its behaviour in a relative time setting by adding the axioms **Until** and **Delay**.

Definition 2.2 (Timed bisimulation)
Let $L_i = (S_i, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_0^i, \longrightarrow_i, \mathcal{U}^i)$, $i \in \{1, 2\}$, be two TTS. A *timed bisimulation* is a relation $R \subseteq S_1 \times S_2$ with $s_0^1 R s_0^2$ satisfying, for all $a(d) \in \mathbf{A} \times \mathbb{R}^{\geq 0}$, the following transfer properties:

1. if $s_1 R s_2$ and $s_1 \xrightarrow{a(d)}_1 s_1'$, then $\exists s_2' \in S_2 :\ s_2 \xrightarrow{a(d)}_2 s_2'$ and $s_1' R s_2'$;
2. if $s_1 R s_2$ and $s_2 \xrightarrow{a(d)}_2 s_2'$, then $\exists s_1' \in S_1 :\ s_1 \xrightarrow{a(d)}_1 s_1'$ and $s_1' R s_2'$; and
3. if $s_1 R s_2$, then $\mathcal{U}_d^1(s_1) \iff \mathcal{U}_d^2(s_2)$.

If such a relation exists, we say that L_1 and L_2 are *timed bisimilar* (notation $L_1 \leftrightarrows L_2$). □

2.3 Timed Automata

In this paragraph we define a variation of timed automata [AD94]. We use invariants as in [HNSY94, NSY92, NSY93] but, instead of considering clock resettings on the edges, we consider them in the states. The reason for this is that we want to avoid assumptions about the initial setting of clocks, which makes the compositionality of the language more complicated. Compare [YPD94] (see Section 8).

Definition 2.3 (Timed automata)
A *timed (safety) automaton* is a structure $(S, \mathbf{A}, \mathcal{C}, s_0, \longrightarrow, \partial, \kappa)$ where:

- S is a set of *states*, with the *initial state* $s_0 \in S$;
- $\mathbf{A}$ is a set of *actions*;
- $\mathcal{C}$ is a set of *clocks*;
- $\longrightarrow \subseteq S \times \mathbf{A} \times \Phi(\mathcal{C}) \times S$ is the set of *edges*;
- $\partial : S \to \overline{\Phi}(\mathcal{C})$ is the *invariant assignment* function;
- $\kappa : S \to \wp_{\text{fin}}(\mathcal{C})$ is the *clocks resettings* function.

The set of all timed automata is denoted by $\mathcal{T}$. □

In this case, $\langle s, a, \phi, s' \rangle \in \longrightarrow$ (notation $s \xrightarrow{a,\phi} s'$) intuitively means that when the system is in state s it could change to be in state s' by performing an action a provided that the clock constraint ϕ holds. The clock setting function states which clocks should be reset as soon as a state is reached. The invariant assignment function states that the system can idle in a state s as long as $\partial(s)$ holds.

Notice that our timed automata can be translated into timed automata with resettings on the edges by just labelling the edge with the set of clocks to be reset in the target state, that is, an edge $s \xrightarrow{a,\phi} s'$ will be translated into $s \xrightarrow{a,\phi,\kappa(s')} s'$. Conversely, a timed automaton with resettings on the edges could be transformed by "pushing" the clock resetting into the target state, i.e., given an edge $s \xrightarrow{a,\phi,C} s'$ we define $s \xrightarrow{a,\phi} s'$ and $\kappa(s') \stackrel{\text{def}}{=} C$. In case that many edges with different clock resettings go to the same state, this state is "split" into different states, one for each set of clocks.

Formally speaking, a timed automaton can be interpreted as a TTS as follows.

Definition 2.4 (Interpretation of timed automata)
Let $T = (S, \mathbf{A}, \mathcal{C}, s_0, \longrightarrow, \partial, \kappa) \in \mathcal{T}$ be a timed automaton. Let $v_0 \in \mathcal{V}^c$ be any valuation. The *interpretation* of T with initial valuation v_0 is given by the TTS $(\!|T|\!)_{v_0} \stackrel{\text{def}}{=} (S \times \mathcal{V}^c, \mathbf{A} \times \mathbb{R}^{\geq 0}, (s_0, v_0), \longrightarrow, \mathcal{U})$ where $\longrightarrow$ and $\mathcal{U}$ are defined as the least sets satisfying the following rules:

$$\frac{s \xrightarrow{a,\phi} s' \quad \models (v[\kappa(s) \hookleftarrow 0] + d)(\phi \wedge \partial(s))}{(s, v) \xrightarrow{a(d)} (s', (v[\kappa(s) \hookleftarrow 0] + d))} \qquad \frac{\models (v[\kappa(s) \hookleftarrow 0] + d)(\partial(s))}{\mathcal{U}_d(s, v)}$$

□

Since $\partial(s) \in \overline{\Phi}(\mathcal{C})$ for all $s \in S$, it follows that $(\!|T|\!)_{v_0}$ satisfies axiom **Until**. Moreover, notice that if $(s, v) \xrightarrow{a(d)}$ then $\models (v[\kappa(s) \hookleftarrow 0] + d)(\partial(s))$ and so $\mathcal{U}_d(s, v)$ which implies that axiom **Delay** holds. Hence, $(\!|T|\!)_{v_0}$ is indeed a TTS for any initial valuation v_0.

Isomorphism is a fine enough equivalence. Thus, proving the existence of an isomorphism is enough to prove that two timed automata are equivalent in coarser equivalences, for instance, timed bisimulation.

Definition 2.5 (Isomorphism of timed automata)
Let $T = (S, \mathbf{A}, \mathcal{C}, s_0, \longrightarrow, \partial, \kappa)$ and $T' = (S', \mathbf{A}, \mathcal{C}, s'_0, \longrightarrow', \partial', \kappa')$ be two timed automata. An *isomorphism* from T to T' is a bijective function $\Gamma : S \to S'$ such that

1. $\Gamma(s_0) = s'_0$,
2. $s \xrightarrow{a,\phi} s' \iff \Gamma(s) \xrightarrow{a,\phi}{}' \Gamma(s')$,
3. $\partial(s) = \partial'(\Gamma(s))$, and
4. $\kappa(s) = \kappa'(\Gamma(s))$.

We say that T and T' are *isomorphic*, notation $T \cong T'$, if there is an isomorphism between T and T'. □

3 A Simple Language for Timed Automata

In this section we introduce a simple language that contains the necessary operators to represent timed automata. We give the semantics of this language in terms of timed automata. Moreover, we show that any timed automaton could be represented by a term of this language if we add guarded recursion over expressions.

3.1 The Language

Definition 3.1 Let $\mathbf{A}$ be a set of actions and let $\mathcal{C}$ be a set of clocks. The language $\mathcal{L}$ is defined according to the following grammar:

$$p ::= \quad \mathbf{stop} \mid a;p \mid \phi{\mapsto}p \mid p+p \mid \{\!|C|\!\}\, p \mid \psi \mathbin{\triangleright\!\!>} p$$

where $a \in \mathbf{A}$, $\phi \in \Phi(\mathcal{C})$, $\psi \in \overline{\Phi}(\mathcal{C})$ and $C \in \wp_{\mathrm{fin}}(\mathcal{C})$. We refer to the elements of $\mathcal{L}$ as processes. □

Process **stop** represents inaction; it is the process that cannot perform any action. The intended meaning of $a;p$ (named *(action-)prefixing*) is that action a can be performed at any time and then it behaves like p. $\phi{\mapsto}p$, the *guarding operation*, executes any first action that p can do whenever ϕ holds. $\{\!|C|\!\}\, p$, the *clock resetting operation*, is a process that behaves like p, but resetting the clocks in C. We will write $\{\!|x_1,\ldots,x_n|\!\}\, p$ instead of $\{\!|\{x_1,\ldots,x_n\}|\!\}\, p$. $\psi \mathbin{\triangleright\!\!>} p$, the *invariant operation*, can idle while ψ holds or go on with the process p. $p+q$ is the *choice*; it executes either p or q. The choice between p and q can be made only by actions, not by the passage of time.

Definition 3.2 (Bound and free variables)
Let $p \in \mathcal{L}$. The set $fv(p)$ of *free variables* of p and the set $bv(p)$ of *bound variables* of p are defined as the least set satisfying

$$\begin{array}{lcllcl}
fv(\mathbf{stop}) & = & \emptyset & bv(\mathbf{stop}) & = & \emptyset \\
fv(a;p) & = & fv(p) & bv(a;p) & = & bv(p) \\
fv(\phi{\mapsto}p) & = & var(\phi) \cup fv(p) & bv(\phi{\mapsto}p) & = & bv(p) \\
fv(p+q) & = & fv(p) \cup fv(q) & bv(p+q) & = & bv(p) \cup bv(q) \\
fv(\{\!|C|\!\}\, p) & = & fv(p) \backslash C & bv(\{\!|C|\!\}\, p) & = & C \cup bv(p) \\
fv(\psi \mathbin{\triangleright\!\!>} p) & = & var(\psi) \cup fv(p) & bv(\psi \mathbin{\triangleright\!\!>} p) & = & bv(p)
\end{array}$$

□

Notice that the term $\{\!|C|\!\}\, p$ binds clocks in C that appear in any constraints in p.

3.2 The Associated Timed Automata

We can associate a timed automaton to a process according to the following definition.

Definition 3.3 (Associated timed automaton)
Let $p \in \mathcal{L}$. ncv, the predicate of *non-conflict of variables* is defined inductively according to rules in Table 1. For all process p such that $ncv(p)$ the *timed automaton associated to* p is defined by $[\![p]\!]^{\mathcal{T}} = (\mathcal{L}, \mathbf{A}, \mathcal{C}, p, \longrightarrow, \partial, \kappa)$ where $\longrightarrow$, ∂ and κ are defined as the least sets satisfying the rules of Table 1. □

Table 1: Timed automata for $\mathcal{L}$

$$ncv(\mathbf{stop}) \qquad \frac{ncv(p) \quad \kappa(p)C \quad (\mathrm{var}(\phi) \cap C = \emptyset)}{ncv(\phi \mapsto p)}$$

$$\frac{ncv(p)}{ncv(a;p)} \qquad \frac{ncv(p) \quad \kappa(p) = C \quad (\mathrm{var}(\psi) \cap C = \emptyset)}{ncv(\psi \rhd\!\!> p)}$$

$$\frac{ncv(p)}{ncv(\{\!|C|\!\}\ p)} \qquad \frac{\begin{array}{lll} ncv(p) & \kappa(p) = C & (C \cap fv(q) = \emptyset) \\ ncv(q) & \kappa(q) = C' & (C' \cap fv(p) = \emptyset) \end{array}}{ncv(p+q)}$$

$$\kappa(\mathbf{stop}) = \emptyset \qquad \kappa(a;p) = \emptyset \qquad \frac{\kappa(p) = C}{\kappa(\phi \mapsto p) = C}$$

$$\frac{\kappa(p) = C \quad \kappa(q) = C'}{\kappa(p+q) = (C \cup C')} \qquad \frac{\kappa(p) = C'}{\kappa(\{\!|C|\!\}\ p) = (C \cup C')} \qquad \frac{\kappa(p) = C}{\kappa(\psi \rhd\!\!> p) = C}$$

$$\partial(\mathbf{stop}) = \mathbf{tt} \qquad \partial(a;p) = \mathbf{tt} \qquad \frac{\partial(p) = \psi}{\partial(\phi \mapsto p) = \psi}$$

$$\frac{\partial(p) = \psi \quad \partial(q) = \psi'}{\partial(p+q) = (\psi \vee \psi')} \qquad \frac{\partial(p) = \psi}{\partial(\{\!|C|\!\}\ p) = \psi} \qquad \frac{\partial(p) = \psi'}{\partial(\psi \rhd\!\!> p) = (\psi \wedge \psi')}$$

$$a;p \xrightarrow{a,\mathbf{tt}} p \qquad \frac{p \xrightarrow{a,\phi'} p'}{\phi \mapsto p \xrightarrow{a,\phi \wedge \phi'} p'}$$

$$\frac{p \xrightarrow{a,\phi} p' \quad \partial(p) = \psi}{\begin{array}{c} p + q \xrightarrow{a,\phi \wedge \psi} p' \\ q + p \xrightarrow{a,\phi \wedge \psi} p' \end{array}} \qquad \frac{p \xrightarrow{a,\phi} p'}{\{\!|C|\!\}\ p \xrightarrow{a,\phi} p'} \qquad \frac{p \xrightarrow{a,\phi} p'}{\psi \rhd\!\!> p \xrightarrow{a,\phi} p'}$$

The next theorem states that the notion of associated timed automaton is well defined for processes without conflict of variables.

Theorem 3.4 *Let $p \in \mathcal{L}$ be a process such that $ncv(p)$. The associated timed automaton $[\![p]\!]^{\mathcal{T}}$ is indeed a timed automaton.*

Proof. It is enough to see that for all $q \in \mathcal{L}$ relations ∂ and κ are functions and moreover, that $\partial(q) \in \overline{\Phi}(\mathcal{C})$. But it can be straightforwardly proven by induction on the depth of the proof tree taking into account that if $\psi, \psi' \in \overline{\Phi}(\mathcal{C})$ then $\psi \wedge \psi', \psi \vee \psi' \in \overline{\Phi}(\mathcal{C})$. □

Rules in Table 1 capture the behaviour described in Section 3.1 in terms of timed automata. Notice that **stop** and $a;p$ have no restriction to idle so $\partial(\mathbf{stop}) = \partial(a;p) = \mathbf{tt}$, moreover they do not reset any clock. As we said above, $a;p$ can perform an action a at any time and then it proceeds with the execution of p. $\phi \mapsto p$ can perform any action p can perform whenever ϕ holds. A process $p+q$ can idle as long as one of them can. Thus $\partial(p+q) \iff \partial(p) \vee \partial(q)$. Moreover $p+q$ can execute any action of p or q as long as it could be executed in its original process. Thus, since an action cannot be executed after the idling time is finished, we require that for the execution of an action, the corresponding invariant must also hold. In principle, processes $\{\!|C|\!\}\ p$, $\psi \circledast p$ can perform any action process p can since these operators only add information to the state. Thus, for $\{\!|C|\!\}\ p$, clocks in C are reset together with the clocks to be reset by p: $\kappa(\{\!|C|\!\}\ p) = \kappa(p) \cup C$. The invariant of $\psi \circledast p$ is restricted to satisfy ψ in addition to the invariant of p, i.e., $\partial(\psi \circledast p) \iff \partial(p) \wedge \psi$.

The condition that processes should not have conflict of variables is necessary. If it were not considered we would have undesirable bindings. For instance, consider the term $p \equiv (x \leq 2) \circledast (\{\!|x|\!\}\ (x = 1) \mapsto a; \mathbf{stop})$. Clearly, x is free in the invariant $(x \leq 2)$, however, using rules in Table 1, we derive $\partial(p) = (x \leq 2)$ and $\kappa(p) = \{x\}$. Thus, according to Definition 2.4 the x in the invariant is captured by the clock resetting. Similar reasoning shows that, in $q \equiv ((y \leq 1) \rhd a; \mathbf{stop}) + (\{\!|y|\!\}\ \mathbf{stop})$, the free occurrence of y in the left operand is captured by the clock resetting in the right operand since $\partial(q) = (y \leq 1)$ and $\kappa(q) = \{y\}$.

One important thing to notice is that the edges preserve the property of non-conflict of variables, that is, if p has no conflict of variables and $p \xrightarrow{a,\phi} p'$ then p' has no conflict of variables. It can be proven by straightforward structural induction.

3.3 Recursion

We extend the expressiveness of our language by allowing recursive specifications.

Definition 3.5 (Recursive specifications)

Let $\mathbf{V}$ be a set of *process variables*. We extend the previous language with process variables. So, let $\mathcal{L}^{\mathbf{v}}$ the language defined by the following grammar:

$$p ::= \mathbf{stop} \mid a;p \mid \phi \mapsto p \mid p+p \mid \{\!|C|\!\}\ p \mid \psi \circledast p \mid X$$

where $a \in \mathbf{A}$, $\phi \in \Phi(\mathcal{C})$, $\psi \in \overline{\Phi}(\mathcal{C})$, $C \subseteq \mathcal{C}$ and $X \in \mathbf{V}$. A *recursive specification* is a set of *recursive equations* having the form

$$X = p(\mathbf{V})$$

for each $X \in \mathbf{V}$, where $p(\mathbf{V}) \in \mathcal{L}^{\mathbf{v}}$. Every recursive specification has a distinguished process variable called *root*. We extend the notion of free and bound variables by adding the equations follows

$$fv(X) = fv(q) \qquad\qquad bv(X) = bv(q)$$

provided $X = q \in E$. fv and bv are then defined by the least sets that satisfy the equations. □

We recall that $fv(p)$ and $bv(p)$ are defined as the least set satisfying the equations in Definitions 3.1 and 3.5. Thus, for instance, if $X = \{\!|x|\!\}\ (y \leq 3 \wedge x < 2) \rhd\!\!\rhd X$ then $fv(X) = \{y\}$ and $bv(X) = \{x\}$.

Now, we extend the notion of the associated timed automaton to recursive specifications and we state the correctness of the definition.

Definition 3.6 (Associated timed automaton)

Let E be a recursive specification such that $ncv(E)$ holds according to rules in Table 1 and Table 2, i.e., E does not have conflict of variables. The *timed automaton associated to* $p \in \mathcal{L}^{\mathbf{v}}$ is defined by $[\![p]\!]^{\mathcal{T}} = (\mathcal{L}, \mathbf{A}, \mathcal{C}, p, \longrightarrow, \partial, \kappa)$ where $\longrightarrow$, ∂ and κ are defined as the least set satisfying rules in Table 1 and rules in Table 2. □

Table 2: Timed automata for recursion

The following rules are defined for all $X = p \in E$

$$ncv(X) \qquad \frac{ncv(p)}{ncv(X = p)} \qquad \frac{\forall X = p \in E.\ ncv(X = p)}{ncv(E)}$$

$$\frac{\kappa(p[p/X]) = C}{\kappa(X) = C} \qquad \frac{\partial(p[p/X]) = \psi}{\partial(X) = \psi} \qquad \frac{p[p/X] \xrightarrow{a,\phi} p'}{X \xrightarrow{a,\phi} p'}$$

Definition 3.7 (Guardedness)

An *occurrence of* X *is guarded* in a term $p \in \mathcal{L}^{\mathbf{v}}$ if p has a subterm $a; q$ such that this occurrence of X is in q. A process variable X is *guarded* in p if every occurrence of it is guarded. A term p is *guarded* if all its variables are guarded. A recursive specification is *guarded* if the right hand side of every recursive equation in it is a guarded process. □

Notice that ∂ and κ are not always well-defined in case of (unguarded!) recursion. For instance, take $X = (x < 1) \rhd\!\!\rhd X$, then ∂ and κ are the completely undefined functions because of nonterminating derivation. Studies on fixed point can be done in these cases. Thus, we would have that $\partial(X) = \psi \wedge (x < 1)$ for all $\psi \in \overline{\Phi}(\mathcal{C})$ and $\kappa(X) = C$ for all $C \in \mathcal{C}$. It seems to be clear that the least fixed point according to set inclusion should be adopted for κ. Therefore, $\kappa(X) = \emptyset$. However, it is not clear which order should be consider for ∂. Compare to the process $X = (x < 1) \rhd\!\!\rhd a; \mathbf{stop} + X$. Nevertheless, we can state the following theorem.

Theorem 3.8 *Let $p \in \mathcal{L}^{\mathbf{v}}$ be a process satisfying $\mathrm{ncv}(p)$, which has process variables defined in a guarded recursive specification E without conflict of variables. The associated timed automaton $[\![p]\!]^{\mathcal{T}}$ is indeed a timed automaton.*

Proof. It can be proved by structural induction that ∂ and κ are defined for any guarded term. In addition, we can see that for all $q \in \mathcal{L}^{\mathbf{v}}$ relations ∂ and κ are functions and moreover, that $\partial(q) \in \overline{\Phi}(\mathcal{C})$. This can be proven by induction on the derivation of ∂ and κ. □

The language presented here, together with a guarded recursive specification, has the property of expressing any timed automaton in the sense of Theorem 3.9 below. First, we borrow some definitions from transition system theory into timed automaton theory. A timed automaton is *image-finite* if the set of outgoing edges of every state labelled with the same action is finite, i.e., for any a and any s, the set $\{s \xrightarrow{a,\phi} s' |\ s' \in S\}$ is finite. It is *finitely sorted* if, for each state s, the set of all actions labelling the outgoing edges, i.e., $\{a |\ \exists s' \in S.\ s \xrightarrow{a,\phi} s'\}$ is finite. A state s is *(symbolically) reachable* if there is a sequence of edges from the initial state s_0 to s, i.e., there are $a_1, \ldots, a_n$, $\phi_1, \ldots, \phi_n$ and $s_1, \ldots, s_n$ $(n \geq 0)$ such that $s_0 \xrightarrow{a_1,\phi_1} s_1 \cdots \xrightarrow{a_n,\phi_n} s_n = s$. The *reachable part* of a timed automaton T is the same timed automaton restricted to the set of states that are reachable. Notice that we are considering a static view but not the usual notion of reachability in timed automata theory (compare to [ACH+92]).

Theorem 3.9 (Representability of timed automata) *For every image-finite and finitely sorted $T \in \mathcal{T}$ there is a guarded recursive specification E with root X_{s_0} such that the reachable part of T and the reachable part of $[\![X_{s_0}]\!]^{\mathcal{T}}$ are isomorphic.*

Proof. The proof consists of associating a process variable to each state s of T and defining each one of them as the term that resets the clocks of $\kappa(s)$ and has an invariant $\partial(s)$ over the summation of the outgoing edges represented by prefixings with its respective guard. Thus, the isomorphism is given by the function that maps every state in its corresponding variable.

Let $T = (S, \mathbf{A}, \mathcal{C}, s_0, \longrightarrow', \partial', \kappa')$. For each state $s \in S$ define a different variable X_s. Let V_S be the set of such variables. Define the set of recursive specifications E with root X_{s_0} and recursive equations

$$X_s = \{\!|\kappa'(s)|\!\}\ \partial'(s) \rhd\!\!\rhd \left(\sum \{\phi \mapsto a; X_{s'} \mid s \xrightarrow{a,\phi} s'\}\right)$$

where $\sum\{p_i \mid i \in \{1 \ldots n\}\} \stackrel{\text{def}}{=} p_1 + p_2 + \cdots + p_n$. In particular, if s has no outgoing transition then

$$X_s = \{\!|\kappa'(s)|\!\}\ \partial'(s) \rhd\!\!\rhd \mathbf{stop}$$

According to Definition 3.6, $[\![X_{s_0}]\!]^{\mathcal{T}} = (\mathcal{L}^{\mathbf{v}}, \mathbf{A}, \mathcal{C}, X_{s_0}, \longrightarrow, \partial, \kappa)$. Define

$$[\![X_{s_0}]\!]^{\mathcal{T}} \restriction V_S \stackrel{\text{def}}{=} (V_S, \mathbf{A}, \mathcal{C}, X_{s_0}, \longrightarrow \restriction V_S, \partial \restriction V_S, \kappa \restriction V_S)$$

where:

$$\begin{array}{lcl} \longrightarrow \restriction V_S & \stackrel{\text{def}}{=} & \longrightarrow \cap (V_S \times \mathbf{A} \times \Phi(\mathcal{C}) \times \mathcal{L}^{\mathbf{v}}) \\ \partial \restriction V_S & \stackrel{\text{def}}{=} & \partial \cap (V_S \times \overline{\Phi}(\mathcal{C})) \\ \kappa \restriction V_S & \stackrel{\text{def}}{=} & \kappa \cap (V_S \times \wp(\mathcal{C})) \end{array}$$

Clearly, $\Gamma : S \to V_S$ defined as $\Gamma(s) \stackrel{\text{def}}{=} X_s$ for all $s \in S$, is an isomorphism, which straightforwardly implies the theorem. □

In the previous proof, the restriction of $[\![X_{s_0}]\!]^{\mathcal{T}}$ to the set V_S is merely formal, and it is due to the fact that the associated timed automaton is defined considering the whole set of terms $\mathcal{L}^{\mathbf{v}}$ instead of the reachables ones.

In order to represent data it could be needed to consider more general timed automata which are not necessarily image finite or finitely sorted. This kind of automata could be represented in the language by defining an infinite summation operator in the expected way. Thus, Theorem 3.9 could be extended to timed automata with denumerable branching and denumerable sorts.

4 An Operational Semantics

In this section we give a semantics for $\mathcal{L}^{\mathbf{v}}$ in terms of TTS. We state that it coincides (modulo timed bisimulation) with the semantics of the associated timed automaton. Moreover, we study α-conversion in order to give semantic to every term.

4.1 The Operational Semantics

Definition 4.1 (Operational semantics of $\mathcal{L}^{\mathbf{v}}$)
Let E be a recursive specification with process variables $\mathbf{V}$. The TTS of a term $p \in \mathcal{L}^{\mathbf{v}}$ with initial valuation $v_0 \in \mathcal{V}^c$ is defined by $(\![p]\!]^*_{v_0} \stackrel{\text{def}}{=} (\mathcal{L}^{\mathbf{v}} \times \mathcal{V}^c, \mathbf{A} \times \mathbb{R}^{\geq 0}, (p, v_0), \longrightarrow, \mathcal{U})$ where $\longrightarrow$ and $\mathcal{U}$ are the least set satisfying rules in Table 3. □

$(\![p]\!]^*_{v_0}$ is well defined, that is, $(\![p]\!]^*_{v_0}$ satisfies axioms **Until** and **Delay**, which can be proven by straightforward induction on the length of the proof tree. Thus,

Theorem 4.2 *For all $p \in \mathcal{L}^{\mathbf{v}}$ and for all closed valuation v_0, $(\![p]\!]^*_{v_0}$ is indeed a* TTS.

Table 3: Operational semantics for $\mathcal{L}^{\mathbf{v}}$ ($a \in \mathbf{A}$, $d \in \mathbb{R}^{\geq 0}$, $v \in \mathcal{V}^c$)

$$\mathcal{U}_d(\mathbf{stop}, v) \qquad \mathcal{U}_d(a; p, v) \qquad \frac{\mathcal{U}_d(p, v)}{\mathcal{U}_d(\phi \mapsto p, v)} \qquad \frac{\mathcal{U}_d(p, v[C \leftarrow 0])}{\mathcal{U}_d(\{\!|C|\!\}\ p, v)}$$

$$\frac{\models (v + d)(\psi) \quad \mathcal{U}_d(p, v)}{\mathcal{U}_d(\psi \rhd\!\!> p, v)} \qquad \frac{\mathcal{U}_d(p, v)}{\mathcal{U}_d(p + q, v) \quad \mathcal{U}_d(q + p, v)}$$

$$(a; p, v) \xrightarrow{a(d)} (p, v + d) \qquad \frac{\models (v + d)(\phi) \quad (p, v) \xrightarrow{a(d)} (p', v')}{(\phi \mapsto p, v) \xrightarrow{a(d)} (p', v')}$$

$$\frac{(p, v[C \leftarrow 0]) \xrightarrow{a(d)} (p', v')}{(\{\!|C|\!\}\ p, v) \xrightarrow{a(d)} (p', v')} \qquad \frac{\models (v + d)(\psi) \quad (p, v) \xrightarrow{a(d)} (p', v')}{(\psi \rhd\!\!> p, v) \xrightarrow{a(d)} (p', v')}$$

$$\frac{(p, v) \xrightarrow{a(d)} (p', v')}{(p + q, v) \xrightarrow{a(d)} (p', v') \quad (q + p, v) \xrightarrow{a(d)} (p', v')}$$

The following rules are defined for all $X = p \in E$

$$\frac{\mathcal{U}_d(p[p/X], v)}{\mathcal{U}_d(X, v)} \qquad \frac{(p[p/X], v) \xrightarrow{a(d)} (p', v')}{(X, v) \xrightarrow{a(d)} (p', v')}$$

The rules in Table 3 express the intended behaviour of each term in terms of TTS. In this case the execution of a transition or the idling time is made concrete. Thus, for instance, process $\phi \mapsto p$ can actually perform any action a that p can perform at time d in the valuation v whenever the condition ϕ holds in the valuation v after d units of time has passed. Or, on the other hand, process $\psi \rhd\!\!> p$ can idle d units of times in a valuation v if p also can idle d units of time, and moreover, condition ψ holds in the valuation v after d units of time.

Now, we extend the notion of timed bisimilarity to the terms in the language.

Definition 4.3 Two terms $p, q \in \mathcal{L}$ are *timed bisimilar*, notation $p \underline{\leftrightarrow} q$, if and only if for all $v_0 \in \mathcal{V}^c$, $(\!|p|\!)^*_{v_0} \underline{\leftrightarrow} (\!|q|\!)^*_{v_0}$. □

Now, we can prove that all the operators of the language preserve timed bisimulation.

Theorem 4.4 $\underline{\leftrightarrow}$ *is a congruence for all operations in* $\mathcal{L}$.

Proof. Suppose $p \mathrel{\underline{\leftrightarrow}} q$ and $p' \mathrel{\underline{\leftrightarrow}} q'$. Hence, for all $v \in \mathcal{V}^c$, there are timed bisimulations R_v and R'_v such that $(p,v)R_v(q,v)$ and $(p',v)R'_v(q',v)$. Then

- $R_v^p \stackrel{\text{def}}{=} \{(a;p,v),(a;q,v)\} \cup (\bigcup_{d\in\mathbb{R}^{\geq 0}} R_{v+d})$,
- $R_v^g \stackrel{\text{def}}{=} \{(\phi\mapsto p,v),(\phi\mapsto q,v)\} \cup R_v$,
- $R_v^+ \stackrel{\text{def}}{=} \{(p+p',v),(q+q',v)\} \cup R_v \cup R'_v$,
- $R_v^c \stackrel{\text{def}}{=} \{(\{\!|C|\!\}\ p,v),(\{\!|C|\!\}\ q,v)\} \cup R_{v[C\leftarrow 0]}$, and
- $R_v^s \stackrel{\text{def}}{=} \{(\psi \rhd\!\!> p,v),(\psi \rhd\!\!> q,v)\} \cup R_v$

are timed bisimulations. The proof of this fact is straightforward. □

4.2 Comparison

So far we stated two ways of interpreting a term in $\mathcal{L}$. Function $(\![\ ]\!)^*$ associates a TTS to each term and closed valuation. On the other hand, a TTS could be associated to every $p \in \mathcal{L}$ in two steps, namely, by associating a timed automaton to p (see Definition 3.3) and then interpreting such a timed automaton in terms of a TTS according to Definition 2.4. Theorem 4.5 states that both way of interpreting a process are equivalent according to timed bisimulation.

Theorem 4.5 *Let E be a guarded recursive specification with process variables $\mathbf{V}$. For every $p \in \mathcal{L}^{\mathbf{v}}$ without conflict of variables and for every closed valuation v_0, $(\![p]\!)^*_{v_0} \mathrel{\underline{\leftrightarrow}} (\!(\llbracket p \rrbracket^{\mathcal{T}})\!)_{v_0}$.*

Proof. Assume $(\![p]\!)^*_{v_0} = (\mathcal{L}^{\mathbf{v}}\times\mathcal{V}^c, \mathbf{A}\times\mathbb{R}^{\geq 0}, (p,v_0), \longrightarrow, \mathcal{U})$ and $(\!(\llbracket p \rrbracket^{\mathcal{T}})\!)_{v_0} = (\mathcal{L}^{\mathbf{v}}\times\mathcal{V}^c, \mathbf{A}\times\mathbb{R}^{\geq 0}, (p,v_0), \longrightarrow', \mathcal{U}')$. We state that $R \stackrel{\text{def}}{=} \{((q,v),(q,\overline{v}))|\ q \in \mathcal{L} \wedge v\restriction fv(q) = \overline{v}\restriction fv(q)\}$, with $v\restriction C \stackrel{\text{def}}{=} v \cap (C \times \mathbb{R}^{\geq 0})$, is a timed bisimulation. Clearly $(p,v_0)R(p,v_0)$. The rest of the proof follows from the next claim

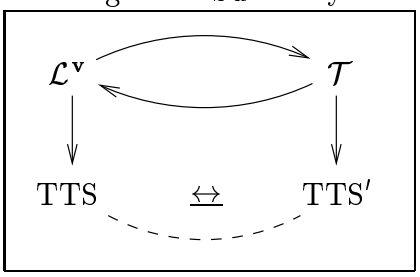
Claim 4.6 *Let $v, \overline{v} \in \mathcal{V}^c$ such that $v\restriction fv(p) = \overline{v}\restriction fv(p)$. Then*

1. *$(p,v) \xrightarrow{a(d)} (p',v')$ implies that $\exists \overline{v}' \in \mathcal{V}^c.\ (p,\overline{v}) \xrightarrow{a(d)}' (p',\overline{v}')$ and $v'\restriction fv(p') = \overline{v}'\restriction fv(p')$*
2. *$(p,\overline{v}) \xrightarrow{a(d)}' (p',\overline{v}')$ implies that $\exists v' \in \mathcal{V}^c.\ (p,v) \xrightarrow{a(d)} (p',v')$ and $v'\restriction fv(p') = \overline{v}'\restriction fv(p')$*
3. *$\mathcal{U}_d(p,v) \iff \mathcal{U}'_d(p,\overline{v})$*

For the proof of the claim see Appendix A. □

A summary of the studied relations is given in Figure 1, where arrows may be read as "can be interpreted in".

Figure 1: Summary



4.3 α-conversion

So far, all the properties were studied for processes without conflict of variables. In this section we show that the behaviour is preserved by α-conversion, which implies that our restriction to the subset of processes without conflict of variables is harmless. We base our studies on [Sto88].

Definition 4.7 (Renaming of clocks in processes)
Let v be a renaming. We extend the notion of *renaming* to terms in $\mathcal{L}$ according to the following recursive definition:

$$\begin{array}{llll} v(\mathbf{stop}) \stackrel{\text{def}}{=} \mathbf{stop} & & v(p+q) \stackrel{\text{def}}{=} v(p)+v(q) \\ v(a;p) \stackrel{\text{def}}{=} a;v(p) & & v(\{\!|C|\!\}\ p) \stackrel{\text{def}}{=} \{\!|f(C)|\!\}\ v[f](p) \\ v(\phi{\mapsto}p) \stackrel{\text{def}}{=} v(\phi){\mapsto}v(p) & & v(\psi \triangleright\!\!> p) \stackrel{\text{def}}{=} v(\psi) \triangleright\!\!> v(p) \end{array}$$

where $f : C \to V$ is a bijective function with $V \in \mathcal{C}$ such that $V \cap v(fv(p) \backslash C) = \emptyset$. □

Definition 4.8 (α-conversion)
Let $\equiv_\alpha \in \mathcal{L} \times \mathcal{L}$ be the least relation satisfying the following rules

$$\mathbf{stop} \equiv_\alpha \mathbf{stop} \qquad \frac{p \equiv_\alpha q}{a;p \equiv_\alpha a;q \quad \phi{\mapsto}p \equiv_\alpha \phi{\mapsto}q \quad \psi \triangleright\!\!> p \equiv_\alpha \psi \triangleright\!\!> q}$$

$$\frac{p \equiv_\alpha q \quad p' \equiv_\alpha q'}{p+p' \equiv_\alpha q+q'} \qquad \frac{f : C \to C' \text{ is bijective} \quad C' \cap fv(\{\!|C|\!\}\ p) = \emptyset \quad \iota[f](p) \equiv_\alpha q}{\{\!|C|\!\}\ p \equiv_\alpha \{\!|C'|\!\}\ q}$$

If $p \equiv_\alpha q$ then p and q are *α-convertibles*. □

It can be proven that $\equiv_\alpha$ is an equivalence, and hence it is a congruence by definition. We refer to [Sto88] for further studies in α-conversion.

In the following we sate that for every term there is an α-conversion which does not have conflict of variables. Together with Theorem 4.10, we can state that for every term, there is another term which is timed bisimilar and does not have conflict of variables. The proof of the following theorem is by straightforward structural induction.

Theorem 4.9 *For every $p \in \mathcal{L}$, there is a $q \in \mathcal{L}$ such that $\mathrm{ncv}(q)$ and $p \equiv_\alpha q$.*

Theorem 4.10 *For all $p, q \in \mathcal{L}$, if $p \equiv_\alpha q$ then $p \underline{\leftrightarrow} q$.*

Proof. We state that

$$R \stackrel{\text{def}}{=} \{((p,v),(q,\overline{v}))|\ \exists \nu.\ \nu \text{ is a renaming} \\ \wedge\, \nu(fv(p)) = fv(q) \wedge \nu(p) \equiv_\alpha q \wedge v\restriction fv(p) = (\overline{v} \circ \nu)\restriction fv(p)\}$$

is a timed bisimulation. Notice that for all $v_0 \in \mathcal{V}^c$, if $p \equiv_\alpha q$ then $(p, v_0)R(q, v_0)$. Thus, proving that R is a timed bisimulation is enough to prove that $p \underline{\leftrightarrow} q$ by Definition 4.3. But the fact that R is a timed bisimulation follows from the next claim.

Claim 4.11 *Assume there exists a renaming ν such that $\nu(fv(p)) = fv(q)$, $\nu(p) \equiv_\alpha q$ and $v\restriction fv(p) = (\overline{v} \circ \nu)\restriction fv(p)$. Then:*

1. *$(p,v) \xrightarrow{a(d)} (p',v')$ implies that exists $(q',\overline{v}')$ such that $(q,\overline{v}) \xrightarrow{a(d)} (q',\overline{v}')$ and $\exists \nu'.\ \nu'$ is a renaming $\wedge\, \nu'(fv(p')) = fv(q') \wedge \nu'(p') \equiv_\alpha q' \wedge v'\restriction fv(p') = (\overline{v}' \circ \nu')\restriction fv(p')$*
2. *$(q,\overline{v}) \xrightarrow{a(d)} (q',\overline{v}')$ implies that exists (p',v') such that $(p,v) \xrightarrow{a(d)} (p',v')$ and $\exists \nu'.\ \nu'$ is a renaming $\wedge\, \nu'(fv(p')) = fv(q') \wedge \nu'(p') \equiv_\alpha q' \wedge v'\restriction fv(p') = (\overline{v}' \circ \nu')\restriction fv(p')$*
3. *$\mathcal{U}_d(p,v)$ iff $\mathcal{U}_d(q,\overline{v})$*

For the proof of the claim see Appendix B. □

Because of Theorems 4.9, 4.5 and 4.10, we can associate a timed automaton to every process in $\mathcal{L}$. We know how to associate a timed automata to processes without conflict of variables. Suppose $p \in \mathcal{L}$ has conflict of variables. Then, we can choose any $q \in \mathcal{L}$ without conflict of variables such that $p \equiv_\alpha q$, and so we define $[\![p]\!]^{\mathcal{T}} \stackrel{\text{def}}{=} [\![q]\!]^{\mathcal{T}}$.

5 Axiomatisation

In this section we give a set of axioms that holds in bisimulation models. It follows immediately that they also hold in any coarser model as for instance the several timed bisimulations with abstraction [Yi90, MT92, Che93, Klu93], timed trace preorder and timed simulations [LV93, LV94]. By Theorems 4.9 and 4.10 we consider terms modulo α-conversion without loss of generality.

5.1 Axioms

Axioms in Table 4 could be explained as follows. The choice is commutative **A1** and associative **A2**. Axioms **A3** and **A3′** state a kind of idempotency of + and **A4** states that **stop** is the neutral element for + in the context of unbounded idling. **Stp** states that a prefixed process which does not satisfies its guard condition cannot proceed with

Table 4: Axioms for $\mathcal{L}$ ($a, b \in \mathbf{A}$, $C \subseteq \mathcal{C}$, $x, y \in \mathcal{C}$, $\phi, \phi' \in \Phi(\mathcal{C})$, $\psi, \psi' \in \overline{\Phi}(\mathcal{C})$, $d \in \mathbb{R}^{\geq 0}$)

A1	$p + q = q + p$	
A2	$(p + q) + r = p + (q + r)$	
A3	$\phi \mapsto p + \phi' \mapsto p = (\phi \vee \phi') \mapsto p$	
A3′	$\psi \boxplus p + \psi' \boxplus p = (\psi \vee \psi') \boxplus p$	
A4	$a; p + \mathbf{stop} = a; p$	
Stp	$\mathbf{ff} \mapsto a; p = \mathbf{stop}$	
G0	$\phi \mapsto \mathbf{stop} = \mathbf{stop}$	
G1	$\mathbf{tt} \mapsto p = p$	
G2	$\phi \mapsto (\phi' \mapsto p) = (\phi \wedge \phi') \mapsto p$	
G3	$\phi \mapsto (\psi \boxplus p) = \psi \boxplus (\phi \mapsto p)$	
G4	$\phi \mapsto (\{\!\{C\}\!\}\ p) = \{\!\{C\}\!\}\ (\phi \mapsto p)$	if $var(\phi) \cap C = \emptyset$
G5	$\phi \mapsto (p + q) = \phi \mapsto p + \phi \mapsto q$	
I1	$\mathbf{tt} \boxplus p = p$	
I2	$\psi \boxplus (\psi' \boxplus p) = (\psi \wedge \psi') \boxplus p$	
I3	$\psi \boxplus (\{\!\{C\}\!\}\ p) = \{\!\{C\}\!\}\ (\psi \boxplus p)$	if $var(\psi) \cap C = \emptyset$
I4	$\psi \boxplus p + \psi \boxplus q = \psi \boxplus (p + q)$	
I5	$\psi \boxplus (\phi \mapsto a; p) + \psi' \boxplus (\phi' \mapsto b; q) = (\psi \vee \psi') \boxplus ((\psi \wedge \phi) \mapsto a; p + (\psi' \wedge \phi') \mapsto b; q)$	
R1	$\{\!\{C\}\!\}\ p = p$	if $C \cap fv(p) = \emptyset$
R2	$\{\!\{C \cup \{y, x\}\}\!\}\ p = \{\!\{C \cup \{y\}\}\!\}\ \iota[x \leftarrow y](p)$	
R3	$\{\!\{C\}\!\}\ \{\!\{C'\}\!\}\ p = \{\!\{C \cup C'\}\!\}\ p$	
R4	$\{\!\{C\}\!\}\ p + \{\!\{C\}\!\}\ q = \{\!\{C\}\!\}\ (p + q)$	
D1	$\phi \mapsto a; (\{\!\{y\}\!\}\ p) = \phi \mapsto a; (\{\!\{y\}\!\}\ (x - y \square d) \boxplus p)$	if $\models (\phi \Rightarrow (x \square d))$ and $x \neq y$
D2	$\phi \mapsto a; p = \phi \mapsto a; ((x - y \square d) \boxplus p)$	if $\models (\phi \Rightarrow (x - y \square d))$ where $\square \in \{\leq, <, \geq, >, =\}$

its execution. Axioms **G0**–**G5** state the way in which guards can be simplifyed. Notice that they cannot be eliminated except in the case of **tt**. In particular, axioms **G3**, **G4** and **G5** say how to move invariants, clock resettings and summations out of the scope of a guard. Similarly, axioms **I1**–**I5** state how to simplify the invariant operation. **I3** says how to take clocks resettings out of the scope of an invariant, while **I4** and **I5** move the invariant out of the scope of a summation. **R1** and **R2** eliminate redundant clocks. In particular, **R2** implies that it is always possible to reduce the amount of clocks to be reset to at most one for each clock resetting operation. **R3** gathers all the clocks resettings in only one operation and **R4** moves clocks out of the scope of a summation. Finally, **D1** and **D2** state that the difference between clocks is invariant and thus it could be "transported" along the execution. In particular, **D1** explains how this difference is stated. Notice that axioms do not necessarily preserve free variables. For instance, **G1** allows us to prove $(x \geq 0) \mapsto p = p$.

Now, we state without proof two derived axioms that can be useful in the following.

Property 5.1 *The following properties can be derived from the axioms.*

$$1.\ \ p + p = p \qquad 2.\ \ \psi \rhd\!\!\rangle\, p = \psi \rhd\!\!\rangle\, (\psi \mapsto p) \qquad 3.\ \ \mathbf{ff} \rhd\!\!\rangle\, \mathbf{stop} + p = p$$

For 2. and 3., induction is also necessary.

Notice that $\mathbf{ff} \rhd\!\!\rangle\, a; p = \mathbf{ff} \rhd\!\!\rangle\, (\mathbf{ff} \mapsto a; p) = \mathbf{ff} \rhd\!\!\rangle\, \mathbf{stop}$ but $\mathbf{ff} \rhd\!\!\rangle\, \mathbf{stop} \not\leftrightarrows \mathbf{stop}$. This is due to the fact that timed bisimulation can model the halting of the progress of time. It could be understood as a broken machine that is not longer allowed to remain in the same state and, simultaneously, has no way to leave such a state, i.e., no action can be performed in order to leave such a state. This phenomenon is known as *time deadlock*. The difference with the ordinary deadlock phenomenon is that a system is in deadlock if it reaches a state that cannot perform any action, but such a state need not have any restrictions on idling, which is the case for time deadlock.

Axioms in Table 4 are sound for timed bisimulation as it is stated as follows.

Theorem 5.2 (Soundness) *For all $p, q \in \mathcal{L}^{\mathbf{v}}$, if $p = q$ is deduced by means of equational reasoning using axioms in Table 4, then $p \leftrightarrows q$.*

Proof. For every axiom $p = q$, we define the relation

$$R \stackrel{\text{def}}{=} \{((p, v), (q, v)) |\ v \in \mathcal{V}^c\} \cup Id$$

except for **R1** for which we define

$$\begin{aligned} R \stackrel{\text{def}}{=}\ & \{((\{\!|C|\!\}\ p, v), (p, v)) |\ v \in \mathcal{V}^c \wedge C \cap fv(p) = \emptyset\} \\ & \cup \{((p, v), (p, \overline{v})) |\ v, \overline{v} \in \mathcal{V}^c \wedge v\restriction fv(p) = \overline{v}\restriction fv(p)\} \end{aligned}$$

for **R2** for which we define

$$\begin{aligned} R \stackrel{\text{def}}{=}\ & \{((\{\!|C \cup \{y, x\}|\!\}\ p, v), (\{\!|C \cup \{y\}|\!\}\ \iota[x \leftarrow y](p), v)) |\ v \in \mathcal{V}^c\} \\ & \cup \{((p, v), (q, \overline{v})) |\ v, \overline{v} \in \mathcal{V}^c \wedge \exists \nu.\ \nu \text{ is a renaming} \\ & \qquad \wedge \nu(fv(p)) = fv(q) \wedge \nu(p) \equiv_\alpha q \wedge v\restriction fv(p) = (\overline{v} \circ \nu)\restriction fv(p)\} \end{aligned}$$

for **D1** for which we define

$$\begin{array}{rcl} R & \stackrel{\text{def}}{=} & \{((\phi{\mapsto}a;(\{\!|y|\!\}\ p),v),(\phi{\mapsto}a;(\{\!|y|\!\}\ (x-y\Box d){\rhd\!\!>}\ p),v))|\ v\in\mathcal{V}^c\} \\ & & \cup\{((\{\!|y|\!\}\ p,v),(\{\!|y|\!\}\ (x-y\Box d){\rhd\!\!>}\ p,v))|\ v\in\mathcal{V}^c\wedge\models v(\phi)\}\cup Id \end{array}$$

and for **D2** for which we define

$$\begin{array}{rcl} R & \stackrel{\text{def}}{=} & \{((\phi{\mapsto}a;p,v),(\phi{\mapsto}a;((x-y\Box d){\rhd\!\!>}\ p),v))|\ v\in\mathcal{V}^c\} \\ & & \cup\{((p,v),((x-y\Box d){\rhd\!\!>}\ p,v))|\ v\in\mathcal{V}^c\wedge\models v(\phi)\}\cup Id \end{array}$$

In every case, R is a timed bisimulation, which proof the theorem. □

5.2 Basic Terms

An interesting property that is derived from these axioms is that every term can be expressed in a normal form.

Definition 5.3 (Basic terms)
Define the set $B\subseteq\mathcal{L}$ of *basic terms* inductively as follows:

- $\mathbf{stop}\in B'$
- $p\in B$, $\phi\in\Phi(\mathcal{C})$ and $a\in\mathbf{A}\quad\Longrightarrow\quad\phi{\mapsto}a;p\in B'$
- $p,q\in B'\quad\Longrightarrow\quad p+q\in B'$
- $p\in B'$, $\psi\in\overline{\Phi}(\mathcal{C})$ and $x\in\mathcal{C}\quad\Longrightarrow\quad\{\!|x|\!\}\ \psi{\rhd\!\!>}\ p\in B$

B' is the set of all terms whose clock resettings and invariants are all within the scope of a prefix construction. Notice that a basic term has the general format (modulo **A1**, **A2**, **A3** and **A4**)

$$p=\{\!|x|\!\}\ \psi{\rhd\!\!>}\left(\sum_{i\in I}\phi_i{\mapsto}a_i;p_i\right)$$

where each p_i is already a basic term. We adopt the convention that

$$\sum_{i\in\emptyset}\phi_i{\mapsto}a_i;p_i=\mathbf{stop}.$$

□

Theorem 5.4 *For every term $p\in\mathcal{L}$ there is a term $q\in B$ such that $p=q$ can be proven by means of axioms in Table 4 and α-conversion.*

Proof. By structural induction.
Case **stop**.

$$\mathbf{stop} \overset{\mathbf{R1},\mathbf{I1}}{=} \{\!|x|\!\}\ \mathbf{tt} \rhd\!\!> \mathbf{stop}$$

Case $a;p$. By induction hypothesis assume p is a basic term. Besides, take a fresh variable x. Then

$$a;p \overset{\mathbf{G1}}{=} \mathbf{tt} \mapsto a;p \overset{\mathbf{R1},\mathbf{I1}}{=} \{\!|x|\!\}\ \mathbf{tt} \rhd\!\!> (\mathbf{tt} \mapsto a;p)$$

Case $\phi \mapsto p$. By induction hypothesis assume

$$p = \{\!|x|\!\}\ \psi \rhd\!\!> \left(\sum_{i \in I} \phi_i \mapsto a_i; p_i\right).$$

where each p_i is already a basic term. Moreover, we can assume that $x \notin var(\phi)$. Then

$$\phi \mapsto p \overset{\mathbf{G4},\mathbf{G3}}{=} \{\!|x|\!\}\ \psi \rhd\!\!> \left(\phi \mapsto \sum_{i \in I} \phi_i \mapsto a_i; p_i\right) \overset{\mathbf{G5},\mathbf{G2}}{=} \{\!|x|\!\}\ \psi \rhd\!\!> \left(\sum_{i \in I} (\phi \wedge \phi_i) \mapsto a_i; p_i\right)$$

Case $p + q$. By induction hypothesis assume

$$p = \{\!|x|\!\}\ \psi \rhd\!\!> \left(\sum_{i \in I} \phi_i \mapsto a_i; p_i\right) \quad \text{and} \quad q = \{\!|y|\!\}\ \psi' \rhd\!\!> \left(\sum_{j \in J} \phi'_j \mapsto b_j; q_j\right).$$

where each p_i and q_j are already basic terms. Moreover, by α-conversion, we can consider $x = y$. Then

$$\begin{aligned}
& p + q \\
\overset{\text{IH}}{=}\ & \{\!|x|\!\}\ \psi \rhd\!\!> \left(\sum_{i \in I} \phi_i \mapsto a_i; p_i\right) + \{\!|x|\!\}\ \psi' \rhd\!\!> \left(\sum_{j \in J} \phi'_j \mapsto b_j; q_j\right) \\
\overset{\mathbf{R4},\mathbf{I4}}{=}\ & \{\!|x|\!\} \left(\sum_{i \in I} \psi \rhd\!\!> (\phi_i \mapsto a_i; p_i) + \sum_{j \in J} \psi' \rhd\!\!> (\phi'_j \mapsto b_j; q_j)\right) \\
\overset{\mathbf{A1},\mathbf{A2},\mathbf{A3}}{=}\ & \{\!|x|\!\} \left(\sum_{i \in I} (\psi \rhd\!\!> (\phi_i \mapsto a_i; p_i) + \psi' \rhd\!\!> (\phi'_1 \mapsto b_1; q_1)) \right. \\
& \qquad \left. + \sum_{j \in J} (\psi' \rhd\!\!> (\phi'_j \mapsto b_j; q_j) + \psi \rhd\!\!> (\phi_1 \mapsto a_1; p_1))\right)
\end{aligned}$$

$$
\begin{aligned}
&\overset{\mathbf{I5}}{=} && \{\!|x|\!\}\ \left(\sum_{i\in I}(\psi\vee\psi')\boxplus((\psi\wedge\phi_i)\mapsto a_i;p_i+(\psi'\wedge\phi'_1)\mapsto b_1;q_1)\right.\\
&&&\left.\quad+\sum_{j\in J}(\psi\vee\psi')\boxplus((\psi'\wedge\phi'_j)\mapsto b_j;q_j+(\psi\wedge\phi_1)\mapsto a_1;p_1)\right)\\
&\overset{\mathbf{I4}}{=} && \{\!|x|\!\}\ (\psi\vee\psi')\boxplus\left(\sum_{i\in I}((\psi\wedge\phi_i)\mapsto a_i;p_i+(\psi'\wedge\phi'_1)\mapsto b_1;q_1)\right.\\
&&&\left.\quad+\sum_{j\in J}((\psi'\wedge\phi'_j)\mapsto b_j;q_j+(\psi\wedge\phi_1)\mapsto a_1;p_1)\right)\\
&\overset{\mathbf{A1},\mathbf{A2},\mathbf{A3}}{=} && \{\!|x|\!\}\ (\psi\vee\psi')\boxplus\left(\sum_{i\in I}(\psi\wedge\phi_i)\mapsto a_i;p_i+\sum_{j\in J}(\psi'\wedge\phi'_j)\mapsto b_j;q_j\right)
\end{aligned}
$$

Case $\{\!|C|\!\}\ p$. By induction hypothesis assume

$$p=\{\!|x|\!\}\ \psi\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right).$$

where each p_i is already a basic term. Then

$$
\begin{aligned}
\{\!|C|\!\}\ p &\overset{\mathrm{IH}}{=} \{\!|C|\!\}\ \{\!|x|\!\}\ \psi\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right)\\
&\overset{\mathbf{R3}}{=} \{\!|C\cup\{x\}|\!\}\ \psi\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right)\\
&\overset{\mathbf{R2}}{=} \{\!|x|\!\}\ (\iota[C\leftarrow x]\psi)\boxplus\left(\sum_{i\in I}(\iota[C\leftarrow x]\phi_i)\mapsto a_i;(\iota[C\leftarrow x]p_i)\right)
\end{aligned}
$$

Case $\psi\boxplus p$. By induction hypothesis assume

$$p=\{\!|x|\!\}\ \psi'\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right).$$

where each p_i is already a basic term. Moreover, we can assume that $x\notin \mathit{var}(\psi)$. Then

$$
\begin{aligned}
\psi\boxplus p &\overset{\mathrm{IH}}{=} \psi\boxplus\{\!|x|\!\}\ \psi'\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right)\\
&\overset{\mathbf{I3},\mathbf{I2}}{=} \{\!|x|\!\}\ (\psi\wedge\psi')\boxplus\left(\sum_{i\in I}\phi_i\mapsto a_i;p_i\right)
\end{aligned}
$$

□

6 Other Operators

In this section, we study several operators. We introduce the hiding operator, the parallel composition and several time operations such as wait, time-out and urgency. In order to simplify the proof of soundness in the following we introduce a strong equivalence over timed automata which implies timed bisimulation.

6.1 Symbolic Bisimulation

We introduce the notion of symbolic bisimulation between timed automata. It is a kind of bisimulation defined directly on timed automata which implies timed bisimulation of the interpreted TTS. The interest of defining this equivalence is not to introduce a new semantic concept but to simplify proofs in the following.

Definition 6.1 (Symbolic bisimulation)

Let $T_i = (S_i, \mathbf{A}, \mathcal{C}, s_0^i, \longrightarrow_i, \partial_i, \kappa_i)$, $i \in \{1,2\}$ be two timed automata. A *symbolic bisimulation* is a relation $R \subseteq S_1 \times S_2$ with $s_0^1 R s_0^2$ satisfying, for all $a \in \mathbf{A}$ and $\phi \in \Phi(\mathcal{C})$, the following transfer properties whenever $s_1 R s_2$

1. $s_1 \xrightarrow{a,\phi}_1 s_1'$, then $\exists s_2' \in S_2 :\ s_2 \xrightarrow{a,\phi'}_2 s_2'$, $\models (\phi \Rightarrow \phi')$ and $s_1' R s_2'$;
2. $s_2 \xrightarrow{a,\phi}_2 s_2'$, then $\exists s_1' \in S_1 :\ s_1 \xrightarrow{a,\phi'}_1 s_1'$, $\models (\phi \Rightarrow \phi')$ and $s_1' R s_2'$;
3. $\models (\partial_1(s_1) \Leftrightarrow \partial_2(s_2))$; and
4. $\kappa_1(s_1) = \kappa_2(s_2)$.

We denote $T_1 \underset{\sim}{\leftrightarrow} T_2$, if there exists a symbolic bisimulation R such that $s_0^1 R s_0^2$. □

Theorem 6.2 *Let $T_1, T_2 \in \mathcal{T}$ such that $T_1 \underset{\sim}{\leftrightarrow} T_2$. Then for all $v_0 \in \mathcal{V}^c$, $(\![T_1]\!)_{v_0} \leftrightarrow (\![T_2]\!)_{v_0}$.*

Proof. Let R be a symbolic bisimulation between T_1 and T_2.

$$R' \stackrel{\text{def}}{=} \{((s_1, v), (s_2, v)) |\ s_1 R s_2\}$$

can be straightforwardly proven to be a timed bisimulation. □

Moreover, notice that $T_1 \cong T_2$ implies $T_1 \underset{\sim}{\leftrightarrow} T_2$. Let $p, q \in \mathcal{L}^{\mathbf{v}}$. In the following we denote $p \underset{\sim}{\leftrightarrow} q$ whenever $[\![p]\!]^{\mathcal{T}} \underset{\sim}{\leftrightarrow} [\![q]\!]^{\mathcal{T}}$.

The notion of symbolic bisimulation up to $\underset{\sim}{\leftrightarrow}$ will simplify considerably several proofs.

Definition 6.3 (Symbolic bisimulation up to $\underset{\sim}{\leftrightarrow}$)

$R \subseteq \mathcal{L}^{\mathbf{v}} \times \mathcal{L}^{\mathbf{v}}$ is a *symbolic bisimulation up to* $\underset{\sim}{\leftrightarrow}$ if and only if pRq implies, for all $a \in \mathbf{A}$ and $\phi \in \Phi(\mathcal{C})$,

1. $p \xrightarrow{a,\phi} p'$, then $\exists q', p'', q'' :\ q \xrightarrow{a,\phi'} q'$, $\models (\phi \Rightarrow \phi')$ and $p' \underset{\sim}{\leftrightarrow} p'' R q'' \underset{\sim}{\leftrightarrow} q'$;
2. $q \xrightarrow{a,\phi} q'$, then $\exists p', p'', q'' :\ p \xrightarrow{a,\phi'} p'$, $\models (\phi \Rightarrow \phi')$ and $p' \underset{\sim}{\leftrightarrow} p'' R q'' \underset{\sim}{\leftrightarrow} q'$;
3. $\models (\partial_1(p) \Leftrightarrow \partial_2(q))$; and
4. $\kappa_1(p) = \kappa_2(q)$. □

Notice that if there is a symbolic bisimulation R up to $\underset{\sim}{\leftrightarrow}$ such that pRq, then it can be proven that $p \underset{\sim}{\leftrightarrow} q$.

6.2 Hiding Operator

We introduce the hiding operator following LOTOS notation [BB89]. In order to do that we introduce a special action $\tau \notin \mathbf{A}$ called *silent action*. The silent action differentiates from the others in the sense that it cannot be observed from the environment. In this work, we are not going to pay special attention on the semantic of such kind of action. $\mathsf{hide}\ A\ \mathsf{in}\ p$ is a process that behaves like p except that all actions in A are renamed into τ. Notice that a renaming operator (see [Mil89, BW90]) can be defined in a similar way. We leave this task for the interested reader.

Define $\mathbf{A}_\tau \stackrel{\text{def}}{=} \mathbf{A} \cup \{\tau\}$. Let $A \subseteq \mathbf{A}$. Free and bounded variables are defined as follows.

$$fv(\mathsf{hide}\ A\ \mathsf{in}\ p) \stackrel{\text{def}}{=} fv(p) \qquad bv(\mathsf{hide}\ A\ \mathsf{in}\ p) \stackrel{\text{def}}{=} bv(p)$$

Rules for the timed automata and TTS are given in Table 5 and Table 6 respectively. The axiomatic definition is given in Table 7.

Table 5: Timed automata for the hiding operator

$$\frac{ncv(p)}{ncv(\mathsf{hide}\ A\ \mathsf{in}\ p)} \qquad \frac{\kappa(p) = C}{\kappa(\mathsf{hide}\ A\ \mathsf{in}\ p) = C} \qquad \frac{\partial(p) = \psi}{\partial(\mathsf{hide}\ A\ \mathsf{in}\ p) = \psi}$$

$$\frac{p \xrightarrow{a,\phi} p'}{\mathsf{hide}\ A\ \mathsf{in}\ p \xrightarrow{a,\phi} \mathsf{hide}\ A\ \mathsf{in}\ p'}\ a \notin A \qquad \frac{p \xrightarrow{a,\phi} p'}{\mathsf{hide}\ A\ \mathsf{in}\ p \xrightarrow{\tau,\phi} \mathsf{hide}\ A\ \mathsf{in}\ p'}\ a \in A$$

Table 6: Operational semantics for the hiding operator

$$\frac{(p,v) \xrightarrow{a(d)} (p',v')}{(\mathsf{hide}\ A\ \mathsf{in}\ p, v) \xrightarrow{a(d)} (\mathsf{hide}\ A\ \mathsf{in}\ p', v')}\ a \notin A \qquad \frac{\mathcal{U}_d(p,v)}{\mathcal{U}_d(\mathsf{hide}\ A\ \mathsf{in}\ p, v)}$$

$$\frac{(p,v) \xrightarrow{a(d)} (p',v')}{(\mathsf{hide}\ A\ \mathsf{in}\ p, v) \xrightarrow{\tau(d)} (\mathsf{hide}\ A\ \mathsf{in}\ p', v')}\ a \in A$$

We extend the definition of renaming and α-conversion according with

$$v(\mathsf{hide}\ A\ \mathsf{in}\ p) \stackrel{\text{def}}{=} \mathsf{hide}\ A\ \mathsf{in}\ v(p) \qquad \frac{p \equiv_\alpha q}{\mathsf{hide}\ A\ \mathsf{in}\ p \equiv_\alpha \mathsf{hide}\ A\ \mathsf{in}\ q}$$

Table 7: Axioms for the hiding operator

$\mathsf{hide}\ A\ \mathsf{in}\ \mathbf{stop} = \mathbf{stop}$	$\mathsf{hide}\ A\ \mathsf{in}\ (\phi \mapsto p) = \phi \mapsto (\mathsf{hide}\ A\ \mathsf{in}\ p)$
$\mathsf{hide}\ A\ \mathsf{in}\ (a;p) = a;(\mathsf{hide}\ A\ \mathsf{in}\ p) \quad \text{if } a \notin A$	$\mathsf{hide}\ A\ \mathsf{in}\ (\{\!\![C]\!\!\}\ p) = \{\!\![C]\!\!\}\ (\mathsf{hide}\ A\ \mathsf{in}\ p)$
$\mathsf{hide}\ A\ \mathsf{in}\ (a;p) = \tau;(\mathsf{hide}\ A\ \mathsf{in}\ p) \quad \text{if } a \in A$	$\mathsf{hide}\ A\ \mathsf{in}\ (\psi \rhd\!\!> p) = \psi \rhd\!\!> (\mathsf{hide}\ A\ \mathsf{in}\ p)$
$\mathsf{hide}\ A\ \mathsf{in}\ (p+q) = \mathsf{hide}\ A\ \mathsf{in}\ p + \mathsf{hide}\ A\ \mathsf{in}\ q$	

Following the same lines of Theorem 4.5, it can be proven that the behaviours of $\mathsf{hide}\ A\ \mathsf{in}\ p$ expressed in the two different ways are equivalent modulo timed bisimulation, i.e., $(\!|\mathsf{hide}\ A\ \mathsf{in}\ p|\!)^*_{v_0} \leftrightarrows (\!|[\![\mathsf{hide}\ A\ \mathsf{in}\ p]\!]^{\mathcal{T}}|\!)_{v_0}$.

In addition, for every term $\mathsf{hide}\ A\ \mathsf{in}\ p$ there is an α-conversion without conflict of variables (see Theorem 4.9). Moreover, Theorem 4.10 still holds if the hiding operator is added, that is, $\equiv_\alpha \subseteq \leftrightarrows$ in the extended language. Besides, $\mathsf{hide}\ A\ \mathsf{in}\ \_$ preserves $\leftrightarrows$.

Theorem 6.4 (Congruence) *If $p \leftrightarrows q$ then $\mathsf{hide}\ A\ \mathsf{in}\ p \leftrightarrows \mathsf{hide}\ A\ \mathsf{in}\ q$.*

Proof. Suppose $p \leftrightarrows q$. Then, for every $v_0 \in \mathcal{V}^c$ there is a timed bisimulation R_{v_0} such that $(p, v_0) R_{v_0} (q, v_0)$. Let $R = \{((\mathsf{hide}\ A\ \mathsf{in}\ p', v), (\mathsf{hide}\ A\ \mathsf{in}\ q', v)) |\ (p', v) R_{v_0} (q', v)\}$ It is easy to prove that R is a timed bisimulation. □

Theorem 6.5 (Soundness) *For all p and q obtained by extending $\mathcal{L}^{\mathbf{v}}$ with the hiding operator, if $p = q$ is deduced by means of equational reasoning using axioms in Table 4 and axioms in Table 7, then $p \leftrightarrows q$.*

Proof. The case of axioms in Table 4 was proven in Theorem 5.2. Let $p = q$ any axiom in Table 7. It is easy to prove that

$$R = \{(p, q)\} \cup Id$$

is a symbolic bisimulation. Now, the theorem follows from Theorem 6.2 and Theorem 4.10. □

Theorem 6.6 (Elimination) *For all term p in the language $\mathcal{L}$ extended with the hiding operator, there is a q in $\mathcal{L}$ such that $p = q$ can be derived from axioms in Table 7.*

Proof. Consider axioms in Table 7 from left to right as rewrite rules. It is simple to prove that the normal form is a term $q \in \mathcal{L}$. □

6.3 Time Operations

In this paragraph we give some axiomatic definitions for common operations on time. The operation $\mathsf{wait}_d(p)$ waits d units of time before starting to execute p. Conversely, $\mathsf{before}_d(p)$ forces to execute p before d units of time have passed. They can be defined as follows, provided $x \notin fv(p)$:

- $\mathsf{wait}_d(p) \stackrel{\text{def}}{=} \{\!|x|\!\}\ (x \geq d) \mapsto p$
- $\mathsf{before}_d(p) \stackrel{\text{def}}{=} \{\!|x|\!\}\ (x \leq d) \rrbracket\!\!> p$

We can modify these operators in order to not include d

- $\mathsf{wait}^{>}_d(p) \stackrel{\text{def}}{=} \{\!|x|\!\}\ (x > d) \mapsto p$
- $\mathsf{before}^{<}_d(p) \stackrel{\text{def}}{=} \{\!|x|\!\}\ (x < d) \rrbracket\!\!> p$

Urgency is defined by the operation $\mathsf{urgent}_d(p)$ that obliges to execute p just after waiting d units of time:

- $\mathsf{urgent}_d(p) \stackrel{\text{def}}{=} \mathsf{before}_d(\,\mathsf{wait}_d(p))$

More generally we can define the operation $\mathsf{between}[d, d'](p)$ which forces the execution of p after waiting d units of time but before d' units of time have passed:

- $\mathsf{between}[d, d'](p) \stackrel{\text{def}}{=} \mathsf{before}_{d'}(\,\mathsf{wait}_d(p))$

We can easily generalise this operation to open intervals in the obvious way.

Maybe, the most well known operation is the time-out. $p\,\mathsf{timeout}_d q$ forces to execute q just after waiting d units of time if process p does not started execution yet:

- $p\,\mathsf{timeout}_d q \stackrel{\text{def}}{=} \mathsf{before}^{<}_d(p) + \mathsf{urgent}_d(q)$

This time-out is called *strong time-out*. the weak version could be defined as:

- $p\,\mathsf{wtimeout}_d q \stackrel{\text{def}}{=} \mathsf{before}_d(p) + \mathsf{urgent}_d(q)$

Consider, for instance, the process $p = \mathsf{wait}_2(\,\mathsf{before}_1(a; \mathbf{stop}))$. p will never perform action a. This fact arises since the clock of the before operator is started together with the clock of the wait operator. Another interpretation for the wait operator is given:

$$\mathsf{wait}'_d(p) = \{\!|x|\!\}\ ((d \leq x) \mapsto \tau; p) \qquad \text{provided } x \notin fv(p).$$

Here, the silent step is used to force the clocks of p to not start.

Table 8: Timed automata for the parallel operator

$$\frac{ncv(p) \quad (bv(p) \cap var(q) = \emptyset)}{ncv(p||_A q) \quad ncv(p|_A q) \quad ncv(p \between_A q)} \qquad \frac{ncv(p)}{ncv(\overline{\mathsf{ck}}(p))}$$

$$\frac{\kappa(p) = C \quad \kappa(q) = C'}{\begin{array}{c}\kappa(p||_A q) = (C \cup C') \\ \kappa(p \between_A q) = (C \cup C') \\ \kappa(p|_A q) = (C \cup C')\end{array}} \qquad \frac{\partial(p) = \psi \quad \partial(q) = \psi'}{\begin{array}{c}\partial(p||_A q) = (\psi \wedge \psi') \\ \partial(p \between_A q) = (\psi \wedge \psi') \\ \partial(p|_A q) = (\psi \wedge \psi')\end{array}}$$

$$\frac{p \xrightarrow{a,\phi} p'}{\begin{array}{l}p||_A q \xrightarrow{a,\phi} p'||_A \overline{\mathsf{ck}}(q) \\ q||_A p \xrightarrow{a,\phi} \overline{\mathsf{ck}}(q)||_A p' \\ p \between_A q \xrightarrow{a,\phi} p'||_A \overline{\mathsf{ck}}(q)\end{array}} \; a \notin A \qquad \frac{p \xrightarrow{a,\phi} p' \quad q \xrightarrow{a,\phi'} q'}{\begin{array}{l}p||_A q \xrightarrow{a,\phi \wedge \phi'} p'||_A q' \\ p|_A q \xrightarrow{a,\phi \wedge \phi'} p'||_A q'\end{array}} \; a \in A$$

$$\kappa(\overline{\mathsf{ck}}(p)) = \emptyset \qquad \frac{\partial(p) = \psi}{\partial(\overline{\mathsf{ck}}(p)) = \psi} \qquad \frac{p \xrightarrow{a,\phi} p'}{\overline{\mathsf{ck}}(p) \xrightarrow{a,\phi} p'}$$

6.4 Parallel Operator

We define a LOTOS-like parallel operator [BB89]. Basically, the process $p||_A q$ executes process p and q in parallel and forces synchronisation on actions in set $A \in \mathbf{A}$. $\between_A$ and $|_A$ are the left and communication merge respectively, which are needed to give a finite axiomatisation of the parallel operator. In order to define associated timed automata we will require the auxiliary operator $\overline{\mathsf{ck}}$ which is intended to avoid clocks resettings.

Free and bound variables are defined by

$$fv(p||_A q) = fv(p \between_A q) = fv(p|_A q) = fv(p) \cup fv(q) \qquad fv(\overline{\mathsf{ck}}(p)) = \kappa(p) \cup fv(p)$$

$$bv(p||_A q) = bv(p \between_A q) = bv(p|_A q) = bv(p) \cup bv(q) \qquad bv(\overline{\mathsf{ck}}(p)) = bv(p)$$

Notice that $bv(\overline{\mathsf{ck}}(p)) = bv(p) \backslash \kappa(p)$ is not generally true. A counterexample is $p \equiv \{\!|x|\!\}\ (x < 1) \mapsto a; \{\!|x|\!\}\ (x < 1) \mapsto a; \mathbf{stop}$. So, for the sake of correctness in our definitions, we choose a wide enough set of bound clocks in $\overline{\mathsf{ck}}(p)$.

We give the rules for the timed automaton in Table 8. Operators $\between_A$ and $|_A$ are the left-merge and the communicating versions of the parallel operator, respectively. Operation $\overline{\mathsf{ck}}$ is needed since if we admitted an edge like $p||_A q \xrightarrow{a,\phi} p'||_A q$ instead of

$p||_A q \xrightarrow{a,\phi} p'||_A \overline{\mathsf{ck}}(q)$, the clocks of q, which were reset as soon as $p||_A q$ was reached, would be reset again when $p'||_A q$ is reached after performing action a. This last situation would be incorrect since the time for process q would then not have progressed.

Axioms for parallel composition are given in Table 9. Operator $\overline{\mathsf{ck}}$ is just required in order to define associated timed automata. Moreover, it does not preserve $\underline{\leftrightarrow}$ and α-conversion. Thus, we are not interested in giving any axiomatisation of it. However, the information introduced for $\overline{\mathsf{ck}}$ is somehow encoded in the axiomatisation by the operator B'. Notice that $\mathsf{B}'(p)$ holds when $p \in B'$ according to Definition 5.3, i.e. whenever no clock resetting or invariant appears out of the scope of a prefixing.

Table 9: Axioms for parallel composition

PC	$p\|\|_A q = p \between_A q + q \between_A p + p\|_A q$	
LM1	$\mathbf{stop} \between_A (\psi \boxplus q) = \psi \boxplus \mathbf{stop}$	if $\mathsf{B}'(q)$
LM2	$a;p \between_A (\psi \boxplus q) = \psi \boxplus \mathbf{stop}$	if $a \in A \wedge \mathsf{B}'(q)$
LM3	$a;p \between_A (\psi \boxplus q) = \psi \boxplus a;(p\|\|_A(\psi \boxplus q))$	if $a \notin A \wedge \mathsf{B}'(q)$
LM4	$(\phi \mapsto p) \between_A q = \phi \mapsto (p \between_A q)$	
LM5	$(p+q) \between_A r = p \between_A r + q \between_A r$	
LM6	$(\{\!\{C\}\!\}\ p) \between_A q = \{\!\{C\}\!\}\ (p \between_A q)$	if $C \cap fv(q) = \emptyset$
LM7	$(\psi \boxplus p) \between_A q = \psi \boxplus (p \between_A q)$	
LM8	$p \between_A \{\!\{C\}\!\}\ q = \{\!\{C\}\!\}\ (p \between_A q)$	if $C \cap fv(p) = \emptyset$
CM0	$p\|_A q = q\|_A p$	
CM1	$\mathbf{stop}\|_A \mathbf{stop} = \mathbf{stop}$	
CM2	$\mathbf{stop}\|_A a;p = \mathbf{stop}$	
CM3	$a;p\|_A a;q = a;(p\|\|_A q)$	if $a \in A$
CM4	$a;p\|_A b;q = \mathbf{stop}$	if $a \neq b \vee a \notin A$
CM5	$\phi \mapsto p\|_A q = \phi \mapsto (p\|_A q)$	
CM6	$(p+q)\|_A r = p\|_A r + q\|_A r$	
CM7	$(\{\!\{C\}\!\}\ p)\|_A q = \{\!\{C\}\!\}\ (p\|_A q)$	if $C \cap fv(q) = \emptyset$
CM8	$(\psi \boxplus p)\|_A q = \psi \boxplus (p\|_A q)$	

UB1	$\mathsf{B}'(\mathbf{stop})$	**UB2**	$\mathsf{B}'(a;p)$
UB3	$\dfrac{\mathsf{B}'(p)}{\mathsf{B}'(\phi \mapsto p)}$	**UB4**	$\dfrac{\mathsf{B}'(p)\ \ \mathsf{B}'(q)}{\mathsf{B}'(p+q)}$

We extend the definition of renaming and α-conversion according with

$$\begin{array}{lcl} v(p||_A q) & \stackrel{\text{def}}{=} & v(p)||_A v(q) \\ v(p \between_A q) & \stackrel{\text{def}}{=} & v(p) \between_A v(q) \\ v(p|_A q) & \stackrel{\text{def}}{=} & v(p)|_A v(q) \end{array} \qquad \frac{p \equiv_\alpha p' \quad q \equiv_\alpha q'}{\begin{array}{c} p||_A q \equiv_\alpha p'||_A q' \\ p \between_A q \equiv_\alpha p' \between_A q' \\ p|_A q \equiv_\alpha p'|_A q' \end{array}}$$

It can be proven that for every term $p||_A q$, $p \between_A q$ and $p|_A q$ there is an α-convertible term without conflict of variables (see Theorem 4.9.) Thus, for terms with conflict of variables we just assume their interpretation is the timed automata of some α-conversion without conflict of variables. Moreover, timed bisimulation is a congruence for $||_A$, $\between_A$ and $|_A$, which is stated in the following.

Theorem 6.7 (Congruence) *Let $p \leftrightarrows p'$ and $q \leftrightarrows q'$. Then, we have $p||_A q \leftrightarrows p'||_A q'$, $p \between_A q \leftrightarrows p' \between_A q'$ and $p|_A q \leftrightarrows p'|_A q'$.*

Proof. First we state the following claim.

Claim 6.8 *Let p and p' be two terms in the extended language. Let $v_0 \in \mathcal{V}^c$. Let R be a timed bisimulation between $(\!|[\![p]\!]^{\mathcal{T}}|\!)_{v_0}$ and $(\!|[\![p']\!]^{\mathcal{T}}|\!)_{v_0}$. Let $V \subseteq \mathcal{C}$ such that $V \cap bv(p) = V \cap bv(p') = \emptyset$. Define*

$$\begin{array}{lcl} \overline{R}_V & \stackrel{\text{def}}{=} & \{((q,\overline{v}),(q',\overline{v}'))| \quad (q,v)R(q',v') \\ & & \qquad \wedge \overline{v}|_{fv(q)} = v|_{fv(q)} \wedge \overline{v}'|_{fv(q')} = v'|_{fv(q')} \wedge \overline{v}|_V = \overline{v}'|_V\} \end{array}$$

Then $\overline{R}_V$ is a timed bisimulation between $(\!|[\![p]\!]^{\mathcal{T}}|\!)_{v_0}$ and $(\!|[\![p']\!]^{\mathcal{T}}|\!)_{v_0}$.

The proof of the claim follows straightforwardly by taking into account Definition 2.4. Besides, notice that if $V \subseteq V'$ then $\overline{R}_{V'} \subseteq \overline{R}_V$, and moreover $\overline{R}_{V'} = \overline{(\overline{R}_V)}_{V'}$.

Now, the theorem follows from this other claim.

Claim 6.9 *Let $\overline{p}$ and $\overline{p}'$ be two terms in the extended language such tha $\overline{p} \leftrightarrows \overline{p}'$. Then, for all $v_0 \in \mathcal{C}$, there is a timed bisimulation R between $(\!|[\![\overline{p}]\!]^{\mathcal{T}}|\!)_{v_0}$ and $(\!|[\![\overline{p}']\!]^{\mathcal{T}}|\!)_{v_0}$. Define:*

$$\begin{array}{lcl} S_1 & \stackrel{\text{def}}{=} & \{((p||_A q, v),(p'||_A q, v'))| \ (p,v)\overline{R}_{\text{var}(q)}(p',v')\} \\ & & \cup\{((\overline{\mathsf{ck}}(p)||_A q, v),(\overline{\mathsf{ck}}(p')||_A q, v'))| \ (p,\overline{v})\overline{R}_{\text{var}(q)}(p',\overline{v}') \\ & & \quad \wedge v\restriction \text{var}(q) = v'\restriction \text{var}(q) \\ & & \quad \wedge \exists d \in \mathbb{R}^{\geq 0}.\ (v\restriction \text{var}(p) = (\overline{v}[\kappa(p)\hookleftarrow 0] + d)\restriction \text{var}(p) \\ & & \qquad \wedge v'\restriction \text{var}(p') = (\overline{v}'[\kappa(p')\hookleftarrow 0] + d)\restriction \text{var}(p'))\} \end{array}$$

$$\begin{array}{lcl} S_1' & \stackrel{\text{def}}{=} & \{((q||_A p, v),(q||_A p', v'))| \ (p,v)\overline{R}_{\text{var}(q)}(p',v')\} \\ & & \cup\{((q||_A \overline{\mathsf{ck}}(p), v),(q||_A \overline{\mathsf{ck}}(p'), v'))| \ (p,v)\overline{R}_{\text{var}(q)}(p',v') \\ & & \quad \wedge v\restriction \text{var}(q) = v'\restriction \text{var}(q) \\ & & \quad \wedge \exists d \in \mathbb{R}^{\geq 0}.\ (v\restriction \text{var}(p) = (\overline{v}[\kappa(p)\hookleftarrow 0] + d)\restriction \text{var}(p) \\ & & \qquad \wedge v'\restriction \text{var}(p') = (\overline{v}'[\kappa(p')\hookleftarrow 0] + d)\restriction \text{var}(p'))\} \end{array}$$

$$S_2 \stackrel{\text{def}}{=} \{((\overline{p}\between_A q, v_0), (\overline{p}'\between_A q, v_0))\} \cup S_1$$

$$S_2' \stackrel{\text{def}}{=} \{((q\between_A \overline{p}, v_0), (q\between_A \overline{p}', v_0))\} \cup S_1'$$

$$S_3 \stackrel{\text{def}}{=} \{((\overline{p}|_A q, v_0), (\overline{p}'|_A q, v_0))\} \cup S_1$$

$$S_3' \stackrel{\text{def}}{=} \{((q|_A \overline{p}, v_0), (q|_A \overline{p}', v_0))\} \cup S_1'$$

All of those relations are timed bisimulation up to $\underline{\leftrightarrow}$.

For the definition of timed bisimulation up to $\underline{\leftrightarrow}$ and the proof of the claim see Appendix C. □

We state that axioms are sound for timed bisimulation and they allow the elimination of these new operators.

Theorem 6.10 (Soundness) *For all p and q obtained by extending $\mathcal{L}^{\mathbf{v}}$ with $||_A$, $\between_A$ and $|_A$, if $p = q$ is deduced by means of equational reasoning using axioms in Table 4 and axioms in Table 9, then $p \underline{\leftrightarrow} q$.*

Proof. The case of axioms in Table 4 was proven in Theorem 5.2.

For axiom **PC** it is routine to prove that

$$\begin{aligned} R \stackrel{\text{def}}{=} & \{((p||_A q, v), (p\between_A q + q\between_A p + p|_A q, v)\} \\ & \cup Id \cup \{((p'||_A q', v), (q'||_A p', v))|\ p' \text{ and } q' \text{ are any term }\} \end{aligned}$$

is a timed bisimulation.

Let $p = q$ any other axiom in Table 9. It is easy to prove that

$$R \stackrel{\text{def}}{=} \{(p, q)\} \cup Id$$

is a symbolic bisimulation except for **LM3** and **LM8** for which it is a symbolic bisimulation up to $\underset{\sim}{\leftrightarrow}$, and for **CM0** for which

$$R \stackrel{\text{def}}{=} \{(p|_A q, q|_A p)\} \cup \{(p'||_A q', q'||_A p')|\ p' \text{ and } q' \text{ are any term }\}$$

could be proven to be a symbolic bisimulation. Now, the theorem follows from Theorem 6.2. □

Theorem 6.11 (Elimination) *For every term p in the language $\mathcal{L}$ extended with $||_A$, $\between_A$ and $|_A$, there is a q in $\mathcal{L}$ such that $p = q$ can be derived from axioms in Table 4 and Table 9.*

Proof. Consider axioms in Table 9 from left to right as rewrite rules modulo axioms in Table 4 and **CM0**. It is simple to prove that the normal form is a term $q \in \mathcal{L}$. □

7 Examples

7.1 The Railroad Crossing

We take the example of the automatic controller of a gate at a railroad crossing using the definition from [AD94], except that we have adapted it to include invariants. The components of the system can be described as follows. A *TRAIN* communicates to the controller that it *appr*oaches at least 2 minutes before it enters the crossing (*in*). After leaving the crossing (*out*), the *TRAIN* informs the *CONTROLLER* that it *exit*ed within 5 minutes after sending the signal *appr*.

The *GATE* system receives the information when to *lower* the gate. This should be put *down* before 1 minute has passed. Then, the system waits for an order to *raise* the gate. After that, it is lifted (*up*) within 1 to 2 minutes.

The *CONTROLLER* waits for a train to *appr*oach. After exactly 1 minute, it orders to *lower* the gate. Then, it waits until the train *exit*s the crossing and at most 1 minute afterwards it orders to *raise* the gate.

The components of the system can be described as follows.

$$
\begin{array}{lcl}
\mathit{TRAIN} & = & \mathit{appr}; \{\!|x|\!\} \quad (\quad (x < 5) \rhd\!\!> (x > 2) \mapsto \mathit{in}; \\
 & & \qquad\qquad\quad (x < 5) \rhd\!\!> \mathit{out}; \\
 & & \qquad\qquad\quad (x < 5) \rhd\!\!> \mathit{exit}; \mathit{TRAIN} \quad) \\
\mathit{GATE} & = & \mathit{lower}; \mathsf{before}_1^{<}(\mathit{down}; \mathit{raise}; \mathsf{between}(1,2)(\mathit{up}; \mathit{GATE})) \\
\mathit{CONTROLLER} & = & \mathit{appr}; \mathsf{urgent}_1(\mathit{lower}; \mathit{exit}; \mathsf{before}_1^{<}(\mathit{raise}; \mathit{CONTROLLER})) \\
\mathit{SYSTEM} & = & \mathit{CONTROLLER} ||_{\{\mathit{appr},\mathit{exit},\mathit{lower},\mathit{raise}\}} (\mathit{TRAIN} ||_{\emptyset} \mathit{GATE})
\end{array}
$$

By using axioms in Table 9 parallel operations can be eliminated. Assuming only one clock for each component, the expression obtained at this point will contain 3 clocks and 19 states. However, many of those states are not reachable since the system will never meet conditions which allow that. These states can be eliminated by using axioms in Table 4, using **D1** and **D2** in particular. Moreover, the number of clocks can be reduced to 2. In this way, the *SYSTEM* can be proven equivalent to the following recursive specification which has 2 clocks and 10 states.

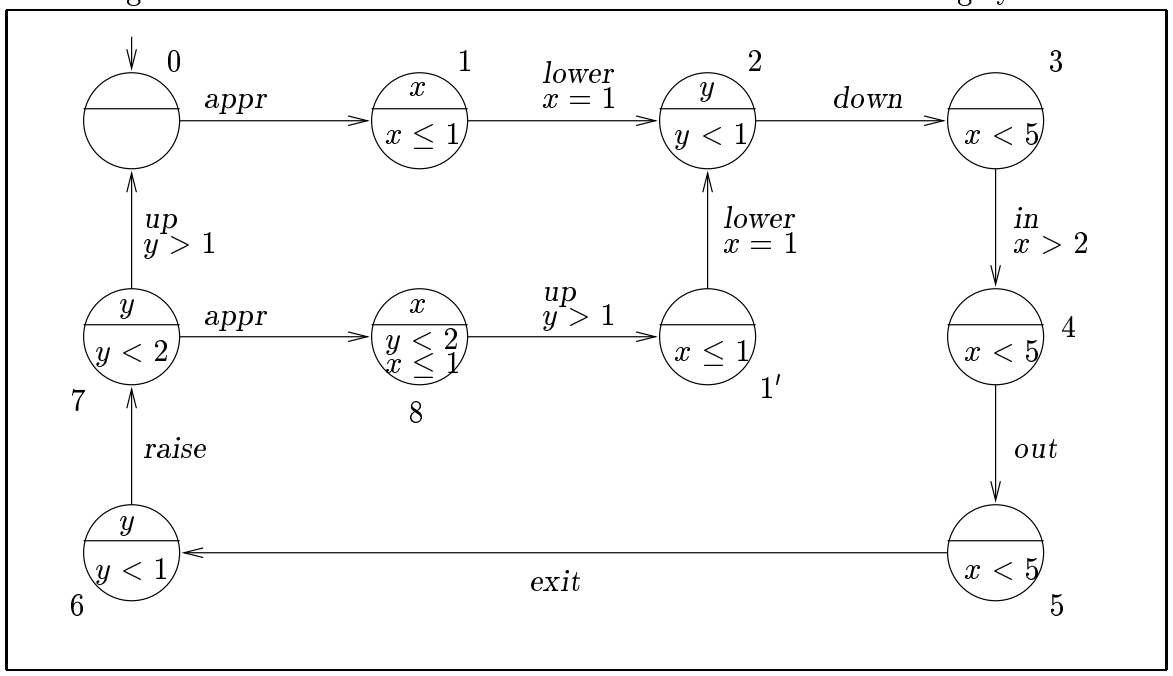
$$
\begin{array}{lcl}
\mathit{SPEC}_0 & = & \mathit{appr}; \mathit{SPEC}_1 \\
\mathit{SPEC}_1 & = & \{\!|x|\!\}\ \mathit{SPEC}_1' \\
\mathit{SPEC}_1' & = & (x \leq 1) \rhd\!\!> (x = 1) \mapsto \mathit{lower}; \mathit{SPEC}_2 \\
\mathit{SPEC}_2 & = & \{\!|y|\!\}\ (y < 1) \rhd\!\!> \mathit{down}; \mathit{SPEC}_3 \\
\mathit{SPEC}_3 & = & (x < 5) \rhd\!\!> (x > 2) \mapsto \mathit{in}; \mathit{SPEC}_4 \\
\mathit{SPEC}_4 & = & (x < 5) \rhd\!\!> \mathit{out}; \mathit{SPEC}_5 \\
\mathit{SPEC}_5 & = & (x < 5) \rhd\!\!> \mathit{exit}; \mathit{SPEC}_6
\end{array}
$$

$$
\begin{aligned}
SPEC_6 &= \{\!|y|\!\}\ (y < 1) \mathbin{\mathbb{I}\!\gg} raise; SPEC_7 \\
SPEC_7 &= \{\!|y|\!\}\ (y < 2) \mathbin{\mathbb{I}\!\gg} (appr; SPEC_8 + (y > 1) \mapsto up; SPEC_0) \\
SPEC_8 &= \{\!|x|\!\}\ (y < 2 \wedge x \leq 1) \mathbin{\mathbb{I}\!\gg} (y > 1) \mapsto up; SPEC_1'
\end{aligned}
$$

Clock x keeps track of the evolution of the time with respect to the *TRAIN* and some activities in the *CONTROLLER* (particularly the action *lower*), while y keeps track of the time of the proper activities of the *GATE* (namely *down* and *up*) and the activity of *rais*ing the gate. Notice, however, that the action *up* in $SPEC_8$ is also constrained by clock x (viz. the condition $x \leq 1$). This would seem to imply that the *CONTROLLER* also controls the time of lifting the gate (action *up*). Clearly, this is not a desirable situation.

The timed automaton associated with $SPEC_0$ is depicted in Figure 2. States are represented by circles and their numbers are written beside. κ and ∂ are respectively written in the upper and lower part of the circle. Edges are represented by the arrows. Empty sets and true conditions are omitted, and singleton sets are represented by their elements.

Figure 2: The reduced timed automaton of the railroad crossing system



7.2 An Improved Version of the Railroad Crossing

We shortly describe the three components of the system. A *TRAIN* communicates to the controller that it *appr*oaches between 3 and 4 minutes before it enters the crossing

(*in*). It takes at most 2 minutes to go *out*. Then it should inform the *CONTROLLER* that it *exit*ed the crossing within 1 minute.

The *GATE* system receives the information when to *lower* the gate. This should be *down* at most 1 minute afterwards. Then, the system waits for an order to *raise* the gate. After that, it is lifted (*up*) within 1 to 2 minutes. If instead, before the gate is *up*, a new order to *lower* the gate is received, then the system does not raise the gate but waits for a new order of *rais*ing it.

The *CONTROLLER* waits for a train to *appr*oach. After exactly 1 minute, it orders to *lower* the gate. Then, it waits until the train *exit*s the crossing and at most 1 minute afterwards it orders to *raise* the gate. However, another *TRAIN* could *appr*oach before the *CONTROLLER* gives such an order. In this case the order of *rais*ing the gate should not be sent; instead, the *CONTROLLER* waits for the train to *exit*.

Thus, each component of the system can be modeled as follows.

$$
\begin{array}{lcl}
TRAIN & = & appr;\ \mathsf{between}[3,4](in;\ \mathsf{before}_2(out;\ \mathsf{before}_1(exit;\ TRAIN))) \\
GATE & = & lower;\ \mathsf{before}_1(down;\ GATE') \\
GATE\text{'} & = & raise;\ (\mathsf{before}_2(lower;\ GATE') + \mathsf{between}(1,2](up;\ GATE)) \\
CONTROLLER & = & appr;\ \mathsf{urgent}_1(lower;\ CONTROLLER') \\
CONTROLLER\text{'} & = & exit;\ \mathsf{before}_1(appr;\ CONTROLLER' \\
 & & \qquad\qquad + raise;\ CONTROLLER) \\
SYSTEM & = & CONTROLLER||_{\{appr,exit,lower,raise\}}(TRAIN||_{\emptyset}GATE)
\end{array}
$$

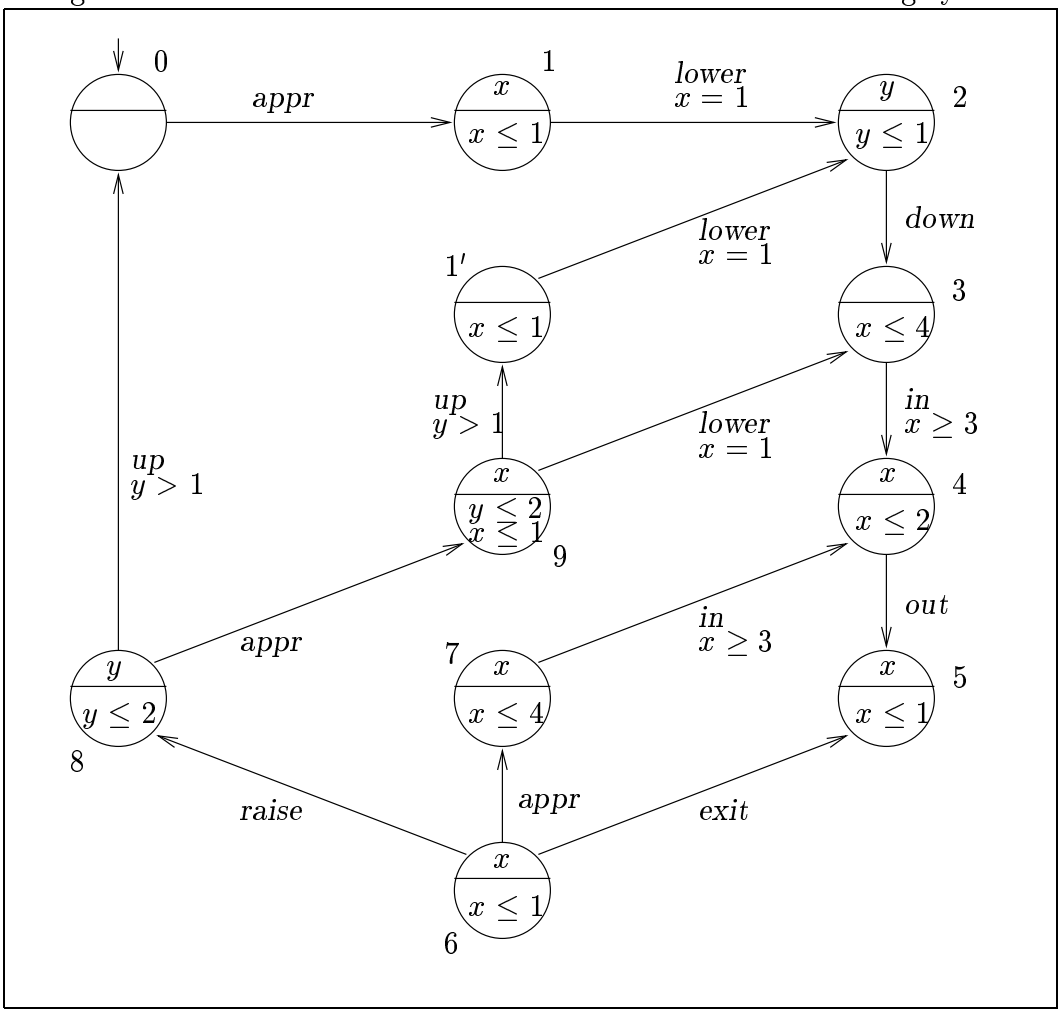
By using axioms in Table 9 parallel operations can be eliminated. Assuming only one clock for each component, the expression obtained at this point will contain 3 clocks and 26 states. However, many of those states are not reachable since the system will never meet conditions which allow that. As before, this states can be eliminated. Thus, the *SYSTEM* can be proven equivalent to the following recursive specification which has 2 clocks and 11 states.

$$
\begin{array}{lcl}
SPEC_0 & = & appr;\ SPEC_1 \\
SPEC_1 & = & \{\!|x|\!\}\ SPEC_1' \\
SPEC_1' & = & (x \le 1) \boxplus\!\!\gg (x = 1) \mapsto lower;\ SPEC_2 \\
SPEC_2 & = & \{\!|y|\!\}\ (y \le 1) \boxplus\!\!\gg down;\ SPEC_3 \\
SPEC_3 & = & (x \le 4) \boxplus\!\!\gg (x \ge 3) \mapsto in;\ SPEC_4 \\
SPEC_4 & = & \{\!|x|\!\}\ (x \le 2) \boxplus\!\!\gg out;\ SPEC_5 \\
SPEC_5 & = & \{\!|x|\!\}\ (x \le 1) \boxplus\!\!\gg exit;\ SPEC_6 \\
SPEC_6 & = & \{\!|x|\!\}\ (x \le 1) \boxplus\!\!\gg (appr;\ SPEC_7 + raise;\ SPEC_8)
\end{array}
$$

$$
\begin{aligned}
SPEC_7 &= \{\!|x|\!\}\ SPEC_3 \\
SPEC_8 &= \{\!|y|\!\}\ (y \leq 2) \boxplus\!\!\gg (appr; SPEC_9 + (y > 1) \mapsto up; SPEC_0) \\
SPEC_9 &= \{\!|x|\!\}\ (y \leq 2 \wedge x \leq 1) \boxplus\!\!\gg ((x = 1) \mapsto lower; SPEC_3 + (y > 1) \mapsto up; SPEC'_1)
\end{aligned}
$$

The timed automaton associated to $SPEC_0$ is depicted in Figure 3.

Figure 3: The reduced timed automaton of the railroad crossing system



7.3 Regions

Using the axioms we can calculate regions. Take for instance the state $SPEC_9$. Notice that it is reached from $SPEC_8$ after performing an *appr* which can be proven to be guarded by $(0 \leq y \leq 2)$. Thus

$(0 \leq y \leq 2) \mapsto appr; SPEC_9 \overset{\mathbf{D1}}{=} (0 \leq y \leq 2) \mapsto appr; \{\!|x|\!\}\ (0 \leq y - x \leq 2) \boxplus\!\!\gg SPEC_9$

So, we calculate

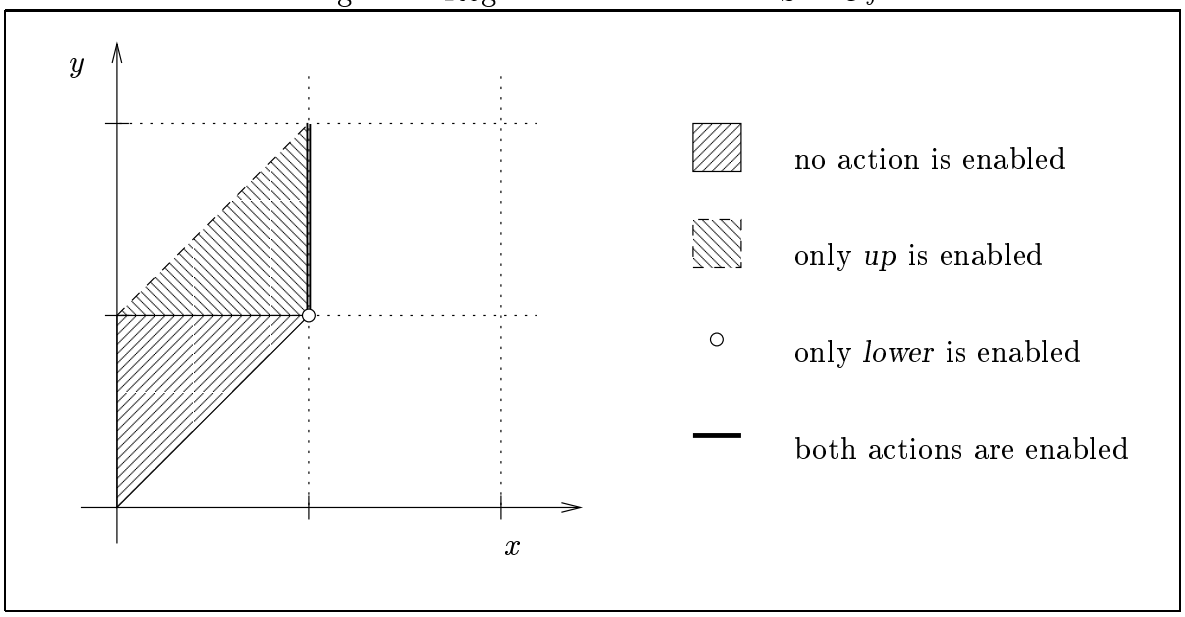
$$\{\!|x|\!\}\ (0 \leq y - x \leq 2) \rhd\!\!\rhd\ SPEC_9$$

$$\overset{\alpha\text{-conv.},\mathbf{I3},\mathbf{R2},\mathbf{I2}}{=} \{\!|x|\!\}\ (y \leq 2 \wedge x \leq 1 \wedge 0 \leq y - x \leq 2) \rhd ((y \leq 2 \wedge x = 1) \mapsto lower; SPEC_3 + (1 < y \leq 2 \wedge x \leq 1) \mapsto up; SPEC_{10})$$

$$\overset{\text{Prop. 5.1.2}}{=} \{\!|x|\!\}\ (x \leq y \leq 2 - x \wedge x \leq 1) \rhd ((1 = x \leq y \leq 2) \mapsto lower; SPEC_3 + (1 < y \leq 2 - x \wedge x \leq 1) \mapsto up; SPEC_{10})$$

Henceforth, we obtained that the system can idle in state $SPEC_9$ whenever $x \leq y \leq 2 - x \wedge x \leq 1$, the action *lower* is enabled in the state $SPEC_9$ whenever $1 = x \leq y \leq 2$, and the action *up* is enabled in the state $SPEC_9$ whenever $1 < y \leq 2 - x \wedge x \leq 1$. This is depicted in Figure 4.

Figure 4: Regions related to state $SPEC_9$



8 Further Remarks

Milner's Synchronisation Trees and our Language

Basically, our calculus is an extension of Milner's synchronisation trees [Mil89] (i.e., CCS with only prefixing, inaction and summation) with operations to manipulate clocks

(clock resettings, invariants and guards). Moreover, we can state that our calculus is an operational conservative extension up to (timed) bisimulation, that is, for every pair of terms obtained by using only prefixing, stop and summation (the *untimed terms*), they are (strong) bisimilar if and only if they are timed bisimilar. It can be easily proven from the following facts.

1. Let $p \in \mathcal{L}$ be an untimed term, let $v, v' \in \mathcal{V}^c$. Then, if there is a $d \in \mathbb{R}^{\geq 0}$ such that $(p, v) \xrightarrow{a(d)} (p', v')$, then for all $d' \in \mathbb{R}^{\geq 0}$ there is a $\overline{v} \in \mathcal{V}^c$ such that $(p, v) \xrightarrow{a(d')} (p', \overline{v})$ and p' is an untimed term. Moreover for all $d \in \mathbb{R}^{\geq 0}$, $\mathcal{U}_d(p, v)$.

2. $p \xrightarrow{a} p'$ if and only if, for all $v \in \mathcal{V}^c$, $(p, v) \xrightarrow{a(d)} (p', v')$ for some $d \in \mathbb{R}^{\geq 0}$ and $v' \in \mathcal{V}^c$.

Furthermore, the equational theory given for $\mathcal{L}$ (see Table 4) is an equational conservative extension of the equational theory for synchronisation trees (i.e. commutativity, associativity, idempotency and **stop** as neutral element of $+$). Thus, for each equality $p = q$ of untimed terms that can be proven in Milner's theory, it can also be proven in our theory and vice versa. These fact can be easily stated. Clearly Milner's axioms can be derived from our theory which proves the "only if". Since our theory is sound and Milner's theory is complete, the "if" follows from the operational conservativity result.

Related Works

Nicollin, Sifakis & Yovine [NSY92, NSY93] give an interpretation of ATP [NS94] in terms of timed automata with invariants, considering a dense time domain. [Yov93] shows that such a translation preserves timed branching bisimulation. ATP is basically an extension of CCS [Mil89] including a timeout operation, an execution delay or *watchdog* operation and the notion of urgent actions. No clocks nor time variables are considered in ATP. Basically the same study was done by *Daws, Olivero & Yovine* [DOY94] for ET-LOTOS [LL94]. In this case, also timed branching bisimulation is shown to be preserved. In neither of these works an inverse study was carried out, i.e., to express a timed automata in terms of the process algebra. In particular, it can be shown that ET-LOTOS is less expressive than $\mathcal{T}$, the set of timed automata.

Fokkink [Fok93, Fok94] sketches an interpretation of ACP with prefix integration [Klu91, Klu93, BB91] into timed automata without invariants. Moreover, the class of strongly regular processes and timed automata turn out to be equivalent when certain restrictions (namely non-Zenoness and fairness) are not present in the behaviour of the timed automata. Thus ACP with prefix integration is more expressive than timed automata. For instance, consider the (finite!) ACP process

$$\int_{v<1} a[v] \cdot \int_{w=v} b[w] \cdot \mathbf{stop}$$

that *records* in v the time when a was performed, and after v units of time executes b. In our language, an unguarded recursive expression would be needed to defined it if the time domain were denumerable. If instead the set of real numbers is considered,

such a process cannot be expressed. However, if we allow more expressive constraints by allowing comparison between clocks, we can define $\{\!|x|\!\}\ (x < 1) \rhd\!\!\gg a; (\{\!|y|\!\}\ (2y \leq x) \rhd\!\!\gg (2y = x) \mapsto b; \mathbf{stop})$. Such an extension would, of course, affect the tractability of the language.

Lynch & Vaandrager [LV94] introduce a language that explicitly manages clocks. Such a language has the same expressive power as timed automata w.r.t. (weak) timed trace equivalence.

Alur & Henzinger [AH94] study the extension of programming languages with clock variables. They discuss their semantics in terms of the so called *real-time programs* [HNSY94] which are easily translated into timed safety automata (see [HNSY94]).

Yi, Pettersson & Daniels [YPD94] give an algebra that represents timed automata without invariants. Basically, the algebra is a syntax for the timed automata including CCS parallel composition and restriction. In particular, the prefixing operation has the form $(\phi, a, C).p$ with $\phi \in \Phi(\mathcal{C})$ and $C \subseteq \mathcal{C}$, and it is the only one that can manage clocks. It could be understood as our term $\phi \mapsto a; \{\!|C|\!\}\ p$. Thus, since terms with conditions in their first actions unavoidably become open terms, it is necessary to consider an initial valuation in its semantics for which $[\mathcal{C} \leftarrow 0]$ is taken. That is rather annoying since even when terms like, for instance, $(x < 1, a, \emptyset).\mathbf{stop}$ and $(y < 1, a, \emptyset).\mathbf{stop}$, show the same behaviour, they become different in the context $(\mathbf{tt}, b, \{x\})._{\_}$. Moreover, notice that this language is strictly less expressive than ours, since it does not include invariant operations.

Conclusions

The contribution of this paper is a language for timed automata. This language is basically an extension of Milner's synchronisation trees with operators to handle clocks, namely clocks resettings, invariants and guards. The language has the ability to represent any (image-finite) timed automata by means of guarded recursion, and moreover, any guarded recursive expression can be interpreted as an (image-finite) timed automata. It is extended with the usual process operations: parallel composition and hiding. Moreover, some common time operations including time-out, waiting and urgency, are algebraically defined in terms of the basic language.

As a secondary goal we introduce a symbolic bisimulation which is not meant to be considered as a proper semantic concept because its discrimination degree, but to conclude, whenever it is possible, timed bisimilarity without the need of using the semantic level, i.e., the timed transition systems.

Also, an equational theory has been given. We have stated that it is sound with respect to timed bisimulation and, moreover, a normal form can be found for each term by using the axioms. With an example we have shown that redundant states, clocks and conditions can be eliminated.

It is interesting to notice that our theory is a conservative extension of Milner's synchronisation trees. We have chosen to use LOTOS-like parallel composition, however, it would also be possible to define the CCS-like parallel composition, restriction and

renaming. In such a case, a conservative extension of the CCS calculus could easily be obtained.

Further study includes reachability analysis by using the equational theory, minimality of clocks according to [HKWT95], completeness of the axiomatisation, particularly whether it is necessary to include an operator like ACP integration [BB91], and axiomatisation of other semantic relations as, for instance, timed trace preorder [LV94].

References

[ACH+92] R. Alur, C. Courcoubetis, N. Halbwachs, D. Dill, and H. Wong-Toi. Minimization of timed transition systems. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 340–354. Springer-Verlag, 1992.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J.Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

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

[AH94] R. Alur and T.A. Henzinger. Real-time system = discrete system + clock variables. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development — Papers presented at First AMAST Workshop on Real-Time System Development,* Iowa City, Iowa, November 1993, pages 1–29. World Scientific, 1994.

[BB89] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. In P.H.L. van Eijk, C.A. Vissers, and M. Diaz, editors, *The formal description technique LOTOS*, pages 23–73. Elsevier Science Publishers, 1989.

[BB91] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Journal of Formal Aspects of Computing Science*, 3(2):142–188, 1991.

[BB95] J.C.M. Baeten and J.A. Bergstra. Real time process algebra with infinitesimals. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes,* Utrecht, 1994, Workshops in Computing, pages 148–187. Springer-Verlag, 1995.

[BK90] J.C.M. Baeten and J.W. Klop, editors. *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BL92] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In de Bakker et al. [dBHRR92], pages 124–148.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Che93] L. Chen. *Timed Processes: Models, Axioms and Decidability.* PhD thesis, Department of Computer Science, University of Edinburgh, 1993.

[Dav92] J. Davies et al. Timed CSP: Theory and practice. In de Bakker et al. [dBHRR92], pages 640–675.

[dBHRR92] J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors. *Proceedings REX Workshop on Real-Time: Theory in Practice,* Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science.* Springer-Verlag, 1992.

[DOY94] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In Hogrefe and Leue [HL94], pages 207–222.

[Fok93] W.J. Fokkink. An elimination theorem for regular behaviours with integration. In E. Best, editor, *Proceedings CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pages 432–446. Springer-Verlag, 1993.

[Fok94] W.J. Fokkink. *Clocks, Trees and Stars in Process Theory.* PhD thesis, University of Amsterdam, December 1994.

[HKWT95] T.A. Henzinger, P.W. Kopke, and H. Wong-Toi. The expressive power of clocks. In P. Wolper, editor, *Proceedings* 22^{th} *ICALP,* Liége, volume 939 of *Lecture Notes in Computer Science*, pages 417–428. Springer-Verlag, 1995. Full version available as Technical Report TR 95-1496, Department of Computer Science, Cornell University, Ithaca, New York, April 1995.

[HL94] D. Hogrefe and S. Leue, editors. *Proceedings of the* 7^{th} *International Conference on Formal Description Techniques, FORTE'94.* North-Holland, 1994.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[Klu91] A.S. Klusener. Completeness in real time process algebra. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 376–392. Springer-Verlag, 1991.

[Klu93] A.S. Klusener. *Models and axioms for a fragment of real time process algebra.* PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, December 1993.

[LL94] G. Leduc and L. Léonard. A formal definition of time in LOTOS. In *Revised draft on enhancements to LOTOS*, 1994. Annex G of document ISO/IEC JTC1/SC21/WG1/Q48.6.

[LV93] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part II: Timing-based systems. Report CS-R9314, CWI, Amsterdam, March 1993. Also, MIT/LCS/TM-487.b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. Submitted. To appear in Information and Computation.

[LV94] N.A. Lynch and F.W. Vaandrager. Action transducers and timed automata. Report CS-R9460, CWI, Amsterdam, November 1994. Also, Technical Memo MIT/LCS/TM-480.b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, October 1994. To appear in *Formal Aspects of Computing.*

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop [BK90], pages 401–415.

[MT92] F. Moller and C. Tofts. Behavioural abstraction in TCCS. In W. Kuich, editor, *Proceedings* 19^{th} *ICALP,* Vienna, volume 623 of *Lecture Notes in Computer Science*, pages 559–570. Springer-Verlag, 1992.

[NS94] X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114(1):131–178, 1994.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE Transactions on Software Engineering*, 18(9):794–804, September 1992.

[NSY93] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. *Acta Informatica*, 30(2):181–202, 1993.

[Sto88] A. Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.

[Yi90] W. Yi. Real-time behaviour of asynchronous agents. In Baeten and Klop [BK90], pages 502–520.

[Yov93] S. Yovine. *Méthodes et outils pour la vérification symbolique de systèmes temporisés.* PhD thesis, Institut National Politechnique de Grenoble, France, May 1993.

[YPD94] W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In Hogrefe and Leue [HL94], pages 223–238.

Appendix

In this appendix we include several proofs with the aim of completeness of this work. We include them appart because of their complexity and length.

A Proof of Claim 4.6

We state the following two proposition without proof

Proposition A.1 *For all* $p, p' \in \mathcal{L}$ *and* $v, v' \in \mathcal{V}^c$, $(p, v) \xrightarrow{a(d)} (p', v')$ *implies* $fv(p') \subseteq fv(p) \cup \kappa(p)$.

Proposition A.2 *For all* $p \in \mathcal{L}^{\mathbf{v}}$ *and* $X = q \in E$, $fv(p) = fv(p[q/X])$

Claim 4.6. *Let* $v, \overline{v} \in \mathcal{V}^c$ *such that* $v \restriction fv(p) = \overline{v} \restriction fv(p)$. *Then*

1. $(p, v) \xrightarrow{a(d)} (p', v')$ *implies that* $\exists \overline{v}' \in \mathcal{V}^c.\ (p, \overline{v}) \xrightarrow{a(d)}{}' (p', \overline{v}')$ *and* $v' \restriction fv(p') = \overline{v}' \restriction fv(p')$
2. $(p, \overline{v}) \xrightarrow{a(d)}{}' (p', \overline{v}')$ *implies that* $\exists v' \in \mathcal{V}^c.\ (p, v) \xrightarrow{a(d)} (p', v')$ *and* $v' \restriction fv(p') = \overline{v}' \restriction fv(p')$
3. $\mathcal{U}_d(p, v) \iff \mathcal{U}'_d(p, \overline{v})$

Proof of the claim.
We prove every case by induction on the depth of the proof tree by considering each case separatedly. Because of Theorem 3.8, κ and ∂ are always defined, so we are no going to remark this fact along the proof.

1.

Case **stop**. This case is trivial since $(\mathbf{stop}, v) \xrightarrow{a(d)}\!\!\!\!\!\!/\ \ $.

Case $a; p$. By Definition 4.1

$(a; p, v) \xrightarrow{a(d)} (p, v + d)$.

Besides, by Definition 3.3,

$a; p \xrightarrow{a, \mathbf{tt}} p, \quad \kappa(a; p) = \emptyset \quad \text{and} \quad \partial(a; p) = \mathbf{tt}$.

Since $\models (\overline{v}[\emptyset \leftarrow 0] + d)(\mathbf{tt} \wedge \mathbf{tt})$, by Definition 2.4,

$(a; p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v} + d)$.

But $v\restriction fv(p) = v\restriction fv(a; p) = \overline{v}\restriction fv(a; p) = \overline{v}\restriction fv(p)$, so $(v + d)\restriction fv(p) = (\overline{v} + d)\restriction fv(p)$ that proves this case.

Case $\phi \mapsto p$. Suppose $(\phi \mapsto p, v) \xrightarrow{a(d)} (p, v')$ By Definition 4.1

$(p, v) \xrightarrow{a(d)} (p, v')$ and $\models (v + d)(\phi)$.

By assumption $v\restriction fv(\phi \mapsto p) = \overline{v}\restriction fv(\phi \mapsto p)$ which implies $v\restriction fv(p) = \overline{v}\restriction fv(p)$. Thus, by induction hypothesis,

$(p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v}')$ and $v'\restriction fv(p') = \overline{v}'\restriction fv(p')$

Moreover, since $v\restriction var(\phi) = \overline{v}\restriction var(\phi)$,

$\models (\overline{v} + d)(\phi)$.

Because of Definition 2.4,

$p \xrightarrow{a,\phi'} p'$ and $\models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\phi' \wedge \partial(p))$,

moreover $\overline{v}' = \overline{v}[\kappa(p) \leftarrow 0] + d$. Thus, by Definition 3.3,

$\phi \mapsto p \xrightarrow{a,\phi \wedge \phi'} p'$.

Since $ncv(\phi \mapsto p)$,

$\models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\phi \wedge \phi' \wedge \partial(p))$.

Hence, by Definition 2.4,

$(\phi \mapsto p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v}[\kappa(p) \leftarrow 0] + d)$

which proves this case.

Case $p + q$. Suppose $(p + q, v) \xrightarrow{a(d)} (p', v')$. By Definition 4.1 suppose

$(p, v) \xrightarrow{a(d)} (p', v')$.

By assumption $v\restriction fv(p + q) = \overline{v}\restriction fv(p + q)$ which implies

$v\restriction fv(p) = \overline{v}\restriction fv(p)$.

Hence, by induction hypothesis

$\exists \overline{v}' \in \mathcal{V}^c.\ (p, \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}')$ and $v'\restriction fv(p') = \overline{v}'\restriction fv(p')$.

Because of Definition 2.4,

$p \xrightarrow{a,\phi} p'$ and $\models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\phi \wedge \partial(p))$,

moreover $\overline{v}' = \overline{v}[\kappa(p) \leftarrow 0] + d$. Since $ncv(p + q)$,

$\models (\overline{v}[\kappa(p + q) \leftarrow 0] + d)(\phi \wedge \partial(p))$.

Hence, taking into account Definition 3.3,

$$p + q \xrightarrow{a,\phi\wedge\partial(p)} p' \quad \text{and} \quad \models (\overline{v}[\kappa(p+q)\leftarrow 0] + d)(\phi \wedge \partial(p) \wedge \partial(p+q)).$$

Thus, by Definition 2.4,

$$(p+q, \overline{v}) \xrightarrow{a(d)}{}' (p', \overline{v}[\kappa(p+q)\leftarrow 0] + d).$$

Besides, since $fv(p') \subseteq fv(p) \cup \kappa(p)$ by Proposition A.1, then $fv(p') \cap \kappa(q) \subseteq \kappa(p)$ because $ncv(p+q)$. Thus

$$\overline{v}'\restriction fv(p') = (\overline{v}[\kappa(p)\leftarrow 0] + d)\restriction fv(p') = (\overline{v}[\kappa(p+q)\leftarrow 0] + d)\restriction fv(p')$$

which prove this case.
The case when $(q, v) \xrightarrow{a(d)} (p', v')$ follows similarly.

Case $\{\!|C|\!\}\ p$. Suppose $(\{\!|C|\!\}\ p, v) \xrightarrow{a(d)} (p', v')$. Then, by Definition 4.1,

$$(p, v[C\leftarrow 0]) \xrightarrow{a(d)} (p', v').$$

Since $v\restriction fv(\{\!|C|\!\}\ p) = \overline{v}\restriction fv(\{\!|C|\!\}\ p)$ and $fv(p) \subseteq C \cup fv(\{\!|C|\!\}\ p)$, then

$$v[C\leftarrow 0]\restriction fv(p) = \overline{v}[C\leftarrow 0]\restriction fv(p).$$

So, by induction hypothesis

$$\exists \overline{v}' \in \mathcal{V}^c.\ (p, \overline{v}[C\leftarrow 0]) \xrightarrow{a(d)}{}' (p', \overline{v}') \text{ and } v'\restriction fv(p') = \overline{v}'\restriction fv(p').$$

Because of Definition 2.4,

$$p \xrightarrow{a,\phi} p' \quad \text{and} \quad \models (\overline{v}[C\leftarrow 0][\kappa(p)\leftarrow 0] + d)(\phi \wedge \partial(p))$$

and moreover, $\overline{v}' = \overline{v}[C\leftarrow 0][\kappa(p)\leftarrow 0] + d = \overline{v}[\kappa(\{\!|C|\!\}\ p)\leftarrow 0] + d$. Now, by Definition 3.3,

$$\{\!|C|\!\}\ p \xrightarrow{a,\phi} p' \quad \text{and} \models (\overline{v}[\kappa(\{\!|C|\!\}\ p)\leftarrow 0] + d)(\phi \wedge \partial(\{\!|C|\!\}\ p))$$

which implies, by Definition 2.4,

$$(\{\!|C|\!\}\ p, \overline{v}) \xrightarrow{a(d)}{}' (p', \overline{v}[\kappa(\{\!|C|\!\}\ p\leftarrow 0] + d)$$

and so, this case is proven.

Case $\psi \rhd\!\!> p$. Suppose $(\psi \rhd\!\!> p, v) \xrightarrow{a(d)} (p', v')$. Then, by Definition 4.1,

$$(p, v) \xrightarrow{a(d)} (p', v') \quad \text{and} \quad \models (v + d)(\psi).$$

Since $v\restriction fv(\psi \rhd\!\!> p) = \overline{v}\restriction fv(\psi \rhd\!\!> p)$ then $v\restriction fv(p) = \overline{v}\restriction fv(p)$ So, by induction hypothesis

$$\exists \overline{v}' \in \mathcal{V}^c.\ (p, \overline{v}) \xrightarrow{a(d)}{}' (p', \overline{v}') \text{ and } v'\restriction fv(p') = \overline{v}'\restriction fv(p').$$

Because of Definition 2.4,

$$p \xrightarrow{a,\phi} p' \quad \text{and} \quad \models (\overline{v}[\kappa(p)\leftarrow 0] + d)(\phi \wedge \partial(p)).$$

Moreover, $\overline{v}' = \overline{v}[\kappa(p)\leftarrow 0] + d$. In addition, because $ncv(\psi \rhd\!\!> p)$,

$$(v + d)\restriction var(\psi) = (\overline{v} + d)\restriction var(\psi) = (\overline{v}[\kappa(p)\leftarrow 0] + d)\restriction var(\psi).$$

That implies

$\models (\overline{v}[\kappa(p)\hookleftarrow 0] + d)(\phi \wedge (\partial(p) \wedge \psi))$.

Thus, by Definition 3.3,

$\psi \rhd\!\!> p \xrightarrow{a,\phi} p'$ and $\models (\overline{v}[\kappa(\psi \rhd\!\!> p)\hookleftarrow 0] + d)(\phi \wedge \partial(\psi \rhd\!\!> p))$

which implies, by Definition 2.4,

$(\psi \rhd\!\!> p, \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}[\kappa(\psi \rhd\!\!> p)\hookleftarrow 0] + d)$

and so, this case is proven.

Case X. Suppose $X = p \in E$ and $(X, v) \xrightarrow{a(d)} (p', v')$. Then, by Definition 4.1,

$(p[p/X], v) \xrightarrow{a(d)} (p', v')$

Taking into account Proposition A.2, $v\restriction fv(p[p/X]) = v\restriction fv(p) = v\restriction fv(X) = \overline{v}\restriction fv(X) = \overline{v}\restriction fv(p) = \overline{v}\restriction fv(p[p/X])$. Thus, by induction hypothesis,

$(p[p/X], \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}')$ and $v'\restriction fv(p') = \overline{v}'\restriction fv(p')$.

Because of Definition 2.4,

$p[p/X] \xrightarrow{a,\phi} p'$ and $\models (\overline{v}[\kappa(p[p/X])\hookleftarrow 0] + d)(\phi \wedge \partial(p[p/X]))$.

Now, by Definition 3.3,

$X \xrightarrow{a,\phi} p'$ and $\models (\overline{v}[\kappa(X)\hookleftarrow 0] + d)(\phi \wedge \partial(X))$

and by Definition 2.4,

$(X, \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}')$

which proves this case.

2.

Case **stop**. This case is trivial since $(\mathbf{stop}, \overline{v}) \xrightarrow{a(d)\!\!\!/} {}'$.

Case $a; p$. Suppose $(a; p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v}')$. By Definition 2.4

$\overline{v}' = \overline{v}[\kappa(a; p)\hookleftarrow 0] + d = \overline{v} + d$.

By Definition 4.1

$(a; p, v) \xrightarrow{a(d)} (p, v + d)$.

Finally, since $v\restriction fv(a; p) = \overline{v}\restriction fv(a; p)$, then $(v + d)\restriction fv(p) = (\overline{v} + d)\restriction fv(p)$ which proves this case.

Case $\phi \mapsto p$. Suppose $(\phi \mapsto p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v}')$. By Definition 2.4

$\phi \mapsto p \xrightarrow{a,\phi\wedge\phi'} p'$ and $\models (\overline{v}[\kappa(\phi \mapsto p)\hookleftarrow 0] + d)(\phi \wedge \phi' \wedge \partial(\phi \mapsto p))$.

Moreover, $\overline{v}' = \overline{v}[\kappa(\phi \mapsto p) \leftarrow\!\!\shortmid 0] + d$. By Definition 3.3,

$$p \xrightarrow{a,\phi'} p', \;\; \kappa(\phi \mapsto p) = \kappa(p) \;\text{ and }\; \partial(\phi \mapsto p) = \partial(p).$$

Thus, $\models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\phi' \wedge \partial(p))$ which implies by Definition 2.4

$$(p, \overline{v}) \xrightarrow{a(d)} {}'(p, \overline{v}').$$

Hence, by induction hypothesis,

$$(p, v) \xrightarrow{a(d)} (p, v') \;\text{ and }\; v' \restriction fv(p') = \overline{v}' \restriction fv(p').$$

Since $ncv(\phi \mapsto p)$, $v \restriction var(\phi) = \overline{v} \restriction var(\phi)$, implying thus

$$\models (v + d)(\phi).$$

So, by Definition 4.1

$$(\phi \mapsto p, v) \xrightarrow{a(d)} (p, v').$$

which proves this case.

Case $p + q$. Suppose $(p + q, \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}')$. By Definition 2.4,

$$p + q \xrightarrow{a,\phi} p' \;\text{ and }\; \models (\overline{v}[\kappa(p + q) \leftarrow\!\!\shortmid 0] + d)(\phi \wedge \partial(p + q)).$$

Moreover, $\overline{v}' = \overline{v}[\kappa(p + q) \leftarrow\!\!\shortmid 0] + d$. By Definition 3.3, suppose

$$p \xrightarrow{a,\phi'} p' \text{ with } \phi = \phi' \wedge \partial(p).$$

Since $ncv(p + q)$,

$$(\overline{v}[\kappa(p + q) \leftarrow\!\!\shortmid 0] + d) \restriction (fv(p) \cup \kappa(p)) = (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d) \restriction (fv(p) \cup \kappa(p))$$

which implies

$$\models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\phi' \wedge \partial(p)).$$

Thus, by Definition 2.4,

$$(p, \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d).$$

Now, by induction hypothesis,

$$\exists v' \in \mathcal{V}^c.\; (p, v) \xrightarrow{a(d)} (p', v') \text{ and } v' \restriction fv(p') = (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d) \restriction fv(p').$$

By Definition 4.1,

$$(p + q, v) \xrightarrow{a(d)} (p', v')$$

Besides, $fv(p') \subseteq fv(p) \cup \kappa(p)$ by Proposition A.1, which implies $fv(p') \cap \kappa(q) \subseteq \kappa(p)$ since $ncv(p + q)$. Thus,

$$v' \restriction fv(p') = (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d) \restriction fv(p') = (\overline{v}[\kappa(p + q) \leftarrow\!\!\shortmid 0] + d) \restriction fv(p'),$$

which proves this case.
The case when $q \xrightarrow{a,\phi'} p'$ is similar.

Case $\{\!|C|\!\}\ p$. Suppose $(\{\!|C|\!\}\ p, \overline{v}) \xrightarrow{a(d)} '(p', \overline{v}')$. By Definition 2.4,

$$\{\!|C|\!\}\ p \xrightarrow{a,\phi} p' \text{ and } \models (\overline{v}[\kappa(\{\!|C|\!\}\ p) \leftarrow 0] + d)(\phi \wedge \partial(\{\!|C|\!\}\ p)).$$

Moreover, $\overline{v}' = \overline{v}[\kappa(\{\!|C|\!\}\ p) \leftarrow 0] + d$. By Definition 3.3,

$$p \xrightarrow{a,\phi} p' \text{ and } \models (\overline{v}[C \cup \kappa(p) \leftarrow 0] + d)(\phi \wedge \partial(p)).$$

Thus, $\models (\overline{v}[C \leftarrow 0][\kappa(p) \leftarrow 0] + d)(\phi \wedge \partial(p))$. Now, by Definition 2.4,

$$(p, \overline{v}[C \leftarrow 0]) \xrightarrow{a(d)} '(p', \overline{v}[C \leftarrow 0][\kappa(p) \leftarrow 0] + d),$$

Since $v \restriction fv(\{\!|C|\!\}\ p) = \overline{v} \restriction fv(\{\!|C|\!\}\ p)$ and $fv(p) \subseteq C \cup fv(\{\!|C|\!\}\ p)$, then $v[C \leftarrow 0] \restriction fv(p) = \overline{v}[C \leftarrow 0] \restriction fv(p)$. Hence, by induction hypothesis,

$$\exists v' \in \mathcal{V}^c.\ (p, v[C \leftarrow 0]) \xrightarrow{a(d)} (p', v') \text{ and } v' \restriction fv(p') = \overline{v}' \restriction fv(p')$$

So, by Definition 4.1,

$$(\{\!|C|\!\}\ p, v) \xrightarrow{a(d)} (p', v').$$

which implies this case.

Case $\psi \rhd\!\!> p$. Suppose $(\psi \rhd\!\!> p, \overline{v}) \xrightarrow{a(d)} '(p', \overline{v}')$. By Definition 2.4,

$$\psi \rhd\!\!> p \xrightarrow{a,\phi} p' \text{ and } \models (\overline{v}[\kappa(\psi \rhd\!\!> p) \leftarrow 0] + d)(\phi \wedge \partial(\psi \rhd\!\!> p)).$$

Moreover, $\overline{v}' = \overline{v}[\kappa(\psi \rhd\!\!> p) \leftarrow 0] + d$. By Definition 3.3,

$$p \xrightarrow{a,\phi} p' \text{ and } \models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\phi \wedge (\partial(p) \wedge \psi)).$$

Thus, $\models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\phi \wedge \partial(p))$. Now, by Definition 2.4,

$$(p, \overline{v}) \xrightarrow{a(d)} '(p', \overline{v}[\kappa(p) \leftarrow 0] + d),$$

Since $v \restriction fv(\psi \rhd\!\!> p) = \overline{v} \restriction fv(\psi \rhd\!\!> p)$, then $v \restriction fv(p) = \overline{v} \restriction fv(p)$. Hence, by induction hypothesis,

$$\exists v' \in \mathcal{V}^c.\ (p, v) \xrightarrow{a(d)} (p', v') \text{ and } v' \restriction fv(p') = \overline{v}' \restriction fv(p')$$

Furthermore, since $ncv(\psi \rhd\!\!> p)$, $\models (\overline{v}[\kappa(p) \leftarrow 0] + d)(\psi)$ implies

$$\models (\overline{v} + d)(\psi).$$

Now, since $v \restriction var(\psi) = \overline{v} \restriction var(\psi)$, then

$$\models (v + d)(\psi).$$

So, by Definition 4.1,

$$(\psi \rhd\!\!> p, v) \xrightarrow{a(d)} (p', v').$$

which implies this case.

Case X. Suppose $X = p \in E$ and $(X, \overline{v}) \xrightarrow{a(d)} '(p', \overline{v}')$. Then, by Definition 2.4,

$$X \xrightarrow{a,\phi} p' \text{ and } \models (\overline{v}[\kappa(X) \leftarrow 0] + d)(\phi \wedge \partial(X)).$$

Now, because of Definition 3.3,

$$p[p/X] \xrightarrow{a,\phi} p' \quad \text{and} \quad \models (\overline{v}[\kappa(p[p/X]) \leftarrow\!\!\shortmid 0] + d)(\phi \wedge \partial(p[p/X]))$$

and by Definition 2.4,

$$(p[p/X], \overline{v}) \xrightarrow{a(d)} {}'(p', \overline{v}')$$

Taking into account Proposition A.2, $\overline{v}\restriction fv(p[p/X]) = \overline{v}\restriction fv(p) = \overline{v}\restriction fv(X) = v\restriction fv(X) = v\restriction fv(p) = v\restriction fv(p[p/X])$. Thus, by induction hypothesis,

$$(p[p/X], v) \xrightarrow{a(d)} (p', v') \text{ and } v'\restriction fv(p') = \overline{v}'\restriction fv(p').$$

So, by Definition 4.1,

$$(X, v) \xrightarrow{a(d)} {}'(p', v')$$

which proves this case.

3.

Case **stop**. For all $v \in \mathcal{V}^c$ and $d \in \mathbb{R}^{\geq 0}$, $\mathcal{U}_d(\mathbf{stop}, v)$ by Definition 4.1. On the other hand, since $\partial(\mathbf{stop}) = \mathbf{tt}$ (see Definition 3.3), and for all $\overline{v} \in \mathcal{V}^c$ and $d \in \mathbb{R}^{\geq 0}$, $\models (\overline{v}[\emptyset \leftarrow\!\!\shortmid 0] + d)(\mathbf{tt})$, $\mathcal{U}'_d(\mathbf{stop}, \overline{v})$ by Definition 2.4. So, for all $v, \overline{v} \in \mathcal{V}^c$, $\mathcal{U}_d(\mathbf{stop}, v) \iff \mathcal{U}'_d(\mathbf{stop}, \overline{v})$.

Case $\phi \mapsto a; p$. By Definition 4.1, for all $v \in \mathcal{V}^c$ and $d \in D$, $\mathcal{U}_d(a; p, v)$. Furthermore, for all $\overline{v} \in \mathcal{V}^c$ and $d \in D$, $\models (\overline{v} + d)(\mathbf{tt})$, which implies, because of Definition 3.3, $\models (\overline{v}[\kappa(a; p) \leftarrow\!\!\shortmid 0] + d)(\partial(a; p))$. So, $\mathcal{U}'_d(a; p, \overline{v})$ by Definition 2.4. Thus $\mathcal{U}_d(a; p, v) \iff \mathcal{U}'_d(a; p, \overline{v})$ for all $v, \overline{v} \in \mathcal{V}^c$.

Case $\phi \mapsto p$. By Definition 4.1, for all $d \in D$,

$$\mathcal{U}_d(\phi \mapsto p, v) \iff \mathcal{U}_d(p, v).$$

Since $v\restriction fv(\phi \mapsto p) = \overline{v}\restriction fv(\phi \mapsto p)$, then $v\restriction fv(p) = \overline{v}\restriction fv(p)$. So, by induction hypothesis,

$$\mathcal{U}_d(p, v) \iff \mathcal{U}_d(p, \overline{v}).$$

Thus, by Definition 2.4,

$$\mathcal{U}_d(p, \overline{v}) \iff \models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\partial(p)).$$

So, because of Definition 3.3,

$$\models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\partial(p) \iff \models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\partial(\phi \mapsto p))$$

and hence, by Definition 2.4,

$$\models (\overline{v}[\kappa(p) \leftarrow\!\!\shortmid 0] + d)(\partial(\phi \mapsto p)) \iff \mathcal{U}'_d(\phi \mapsto p, v)$$

Case $p + q$. By Definition 4.1,

$$\mathcal{U}_d(p + q, v) \iff \mathcal{U}_d(p, v) \vee \mathcal{U}_d(q, v).$$

Since $fv(p) \subseteq fv(p + q)$ and $fv(q) \subseteq fv(p + q)$, by induction hypothesis,

$\mathcal{U}_d(p,v) \vee \mathcal{U}_d(q,v) \iff \mathcal{U}'_d(p,\overline{v}) \vee \mathcal{U}'_d(q,\overline{v})$.

By Definition 2.4,

$\mathcal{U}'_d(p,\overline{v}) \vee \mathcal{U}'_d(q,\overline{v}) \iff \models (\overline{v}+d)(\partial(p)) \vee \models (\overline{v}+d)(\partial(q))$.

Because of Definition 3.3,

$\models (\overline{v}+d)(\partial(p)) \vee \models (\overline{v}+d)(\partial(q)) \iff \models (\overline{v}+d)(\partial(p+q))$.

Finally, by Definition 2.4,

$\models (\overline{v}+d)(\partial(p+q)) \iff \mathcal{U}'_d(p+q,\overline{v})$.

Case $\{\!|C|\!\}\ p$. Because of Definition 4.1,

$\mathcal{U}_d(\{\!|C|\!\}\ p, v) \iff \mathcal{U}_d(p, v[C{\hookleftarrow}0])$.

Since $fv(p) \cup var(\psi) \subseteq fv(\triangleright\!\!\gg C, \psi p) \cup C$, then by induction hypothesis,

$\mathcal{U}_d(p, v[C{\hookleftarrow}0]) \iff \wedge\mathcal{U}'_d(p, \overline{v}[C{\hookleftarrow}0])$.

By Definition 2.4,

$\mathcal{U}'_d(p, \overline{v}[C{\hookleftarrow}0]) \iff \models (\overline{v}[C{\hookleftarrow}0][\kappa(p){\hookleftarrow}0] + d)(\partial(p))$.

By Definition 3.3,

$\models (\overline{v}[C \cup \kappa(p){\hookleftarrow}0] + d)(\partial(p)) \iff \models (\overline{v}[\kappa(\{\!|C|\!\}\ p){\hookleftarrow}0] + d)(\partial(\triangleright\!\!\gg C, \psi p))$.

Finally, by Definition 2.4,

$\models (\overline{v}[\kappa(\{\!|C|\!\}\ p){\hookleftarrow}0] + d)(\partial(\{\!|C|\!\}\ p)) \iff \mathcal{U}'_d(\{\!|C|\!\}\ p, \overline{v})$.

Case $\psi \triangleright\!\!\gg p$. Because of Definition 4.1,

$\mathcal{U}_d(\psi \triangleright\!\!\gg p, v) \iff \models (v+d)(\psi) \wedge \mathcal{U}_d(p,v)$.

Since $v{\restriction}\psi \triangleright\!\!\gg fv(p) = {\restriction}fv(\psi \triangleright\!\!\gg p)$, then $v{\restriction}var(\psi) = \overline{v}{\restriction}var(\psi)$. So, by induction hypothesis,

$\models (v+d)(\psi) \wedge \mathcal{U}_d(p,v) \iff \models (\overline{v}+d)(\psi) \wedge \mathcal{U}'_d(p,\overline{v})$.

By Definition 2.4,

$\models (\overline{v}+d)(\psi) \wedge \mathcal{U}'_d(p,\overline{v}) \iff \models (\overline{v}+d)(\psi) \wedge \models (\overline{v}[\kappa(p){\hookleftarrow}0]+d)(\partial(p))$.

Since $ncv(\psi \triangleright\!\!\gg p)$, $\overline{v}{\restriction}var(\psi) = \overline{v}[\kappa(p){\hookleftarrow}0]{\restriction}var(\psi)$. Thus,

$\models (\overline{v}+d)(\psi) \wedge \models (\overline{v}[\kappa(p){\hookleftarrow}0]+d)(\partial(p)) \iff \models (\overline{v}[\kappa(p){\hookleftarrow}0]+d)(\partial(p) \wedge \psi)$.

By Definition 3.3,

$\models (\overline{v}[\kappa(p){\hookleftarrow}0]+d)(\partial(p) \wedge \psi) \iff \models (\overline{v}[\kappa(\psi \triangleright\!\!\gg p){\hookleftarrow}0]+d)(\partial(\psi \triangleright\!\!\gg p))$.

Finally, by Definition 2.4,

$\models (\overline{v}[\kappa(\psi \triangleright\!\!\gg p){\hookleftarrow}0]+d)(\partial(\psi \triangleright\!\!\gg p)) \iff \mathcal{U}'_d(\psi \triangleright\!\!\gg p, \overline{v})$.

Case X. Suppose $X = p \in E$. Because of Definition 4.1,

$$\mathcal{U}_d(X, v) \iff \mathcal{U}_d(p[p/X], v).$$

Because of Proposition A.2, $v\restriction fv(p[p/X]) = v\restriction fv(p) = v\restriction fv(X) = \overline{v}\restriction fv(X) = \overline{v}\restriction fv(p) = \overline{v}\restriction fv(p[p/X])$. So, by induction hypothesis,

$$\mathcal{U}_d(p[p/X], v) \iff \mathcal{U}'_d(p[p/X], \overline{v}).$$

Now, by Definition 2.4,

$$\mathcal{U}'_d(p[p/X], \overline{v}) \iff \models (\overline{v}[\kappa(p[p/X]) \hookleftarrow 0] + d)(\partial(p[p/X]))$$

and by Definition 3.3,

$$\models (\overline{v}[\kappa(p[p/X]) \hookleftarrow 0] + d)(\partial(p[p/X])) \iff \models (\overline{v}[\kappa(X) \hookleftarrow 0] + d)(\partial(X)).$$

Finally, because of Definition 2.4,

$$\models (\overline{v}[\kappa(X) \hookleftarrow 0] + d)(\partial(X)) \iff \mathcal{U}'_d(X, \overline{v}). \qquad \square$$

B Proof of Claim 4.11

Claim 4.11. *Assume there exists a renaming* ν *such that* $\nu(fv(p)) = fv(q)$, $\nu(p) \equiv_\alpha q$ *and* $v\restriction fv(p) = (\overline{v} \circ \nu)\restriction fv(p)$. *Then:*

1. $(p, v) \xrightarrow{a(d)} (p', v')$ *implies that exists* $(q', \overline{v}')$ *such that* $(q, \overline{v}) \xrightarrow{a(d)} (q', \overline{v}')$ *and* $\exists \nu'.\ \nu'$ *is a renaming* $\wedge\ \nu'(fv(p')) = fv(q') \wedge \nu'(p') \equiv_\alpha q' \wedge v'\restriction fv(p') = (\overline{v}' \circ \nu')\restriction fv(p')$

2. $(q, \overline{v}) \xrightarrow{a(d)} (q', \overline{v}')$ *implies that exists* (p', v') *such that* $(p, v) \xrightarrow{a(d)} (p', v')$ *and* $\exists \nu'.\ \nu'$ *is a renaming* $\wedge\ \nu'(fv(p')) = fv(q') \wedge \nu'(p') \equiv_\alpha q' \wedge v'\restriction fv(p') = (\overline{v}' \circ \nu')\restriction fv(p')$

3. $\mathcal{U}_d(p, v)$ *iff* $\mathcal{U}_d(q, \overline{v})$

Proof of the claim.
1. By structural induction on p.

Suppose $p \equiv \mathbf{stop}$. This case is trivial.

Suppose $p \equiv a; p'$. By the hypothesis and considering Definition 4.7, there exists a renaming ν such that $\nu(fv(a; p')) = fv(q)$ and

$$a; \nu(p') \equiv_\alpha q \wedge v\restriction fv(a; p') = (\overline{v} \circ \nu)\restriction fv(a; p'). \tag{1}$$

So, by Definition 4.8,

$$q \equiv a; q' \text{ with } \nu(p') \equiv_\alpha q'. \tag{2}$$

Thus, by Definition 4.1,

$$(a; q', \overline{v}) \xrightarrow{a(d)} (q', \overline{v} + d).$$

We know that $fv(a; p') = fv(p')$. Now, choose $\nu' = \nu$. So, from (1) and (2) we have

$\exists \nu'.\ \nu'$ is a renaming $\wedge\ \nu'(fv(p')) = fv(q') \wedge \nu'(p') \equiv_\alpha q' \wedge v' \restriction fv(p') = (\overline{v}' \circ \nu') \restriction fv(p')$

that proves this case.

Suppose $p \equiv \phi \mapsto p'$. By the hypothesis and considering Definition 4.7, there exists a renaming ν such that $\nu(fv(\phi \mapsto p')) = fv(q)$ and

$$\nu(\phi) \mapsto \nu(p') \equiv_\alpha q \ \wedge\ v \restriction fv(\phi \mapsto p') = (\overline{v} \circ \nu) \restriction fv(\phi \mapsto p'). \tag{3}$$

So, by Definition 4.8,

$q \equiv \phi' \mapsto q'$ with $\nu(\phi) = \phi' \ \wedge\ \nu(p') \equiv_\alpha q'$.

Suppose that

$(\phi \mapsto p', v) \xrightarrow{a(d)} (p'', v')$.

By Definition 4.1,

$(p', v) \xrightarrow{a(d)} (p'', v')$ and $\models (v + d)(\phi)$.

Because $fv(p') \subseteq fv(p)$, by induction hypothesis, there exists $(q'', \overline{v}')$ such that

$$\begin{aligned}(q', \overline{v}) \xrightarrow{a(d)} (q'', \overline{v}') \text{ and} \\ \exists \nu'.\ \nu' \text{ is a renaming} \\ \wedge\ \nu'(fv(p'')) = fv(q'') \wedge \nu'(p'') \equiv_\alpha q'' \wedge v' \restriction fv(p'') = (\overline{v}' \circ \nu') \restriction fv(p'').\end{aligned} \tag{4}$$

Considering (3), for all $d \in \mathbb{R}^{\geq 0}$,

$(v + d) \restriction fv(p) = ((\overline{v} \circ \nu) + d) \restriction fv(p) = ((\overline{v} + d) \circ \nu) \restriction fv(p)$.

Because of that,

$((\overline{v} + d) \circ \nu)(\phi) = (\overline{v} + d)(\nu(\phi)) = (\overline{v} + d)(\phi')$.

Thus,

$\models (\overline{v} + d)(\phi')$

So, together with (4), by Definition 4.1,

$(\phi' \mapsto q', \overline{v}) \xrightarrow{a(d)} (q'', \overline{v}')$

which proves this case.

Suppose $p \equiv p' + p''$. By the hypothesis and considering Definition 4.7, there exists a renaming ν such that $\nu(fv(p' + p'')) = fv(q)$ and

$$\nu(p') + \nu(p'') \equiv_\alpha q \wedge v \restriction fv(p' + p'') = (\overline{v} \circ \nu) \restriction fv(p' + p''). \tag{5}$$

So, by Definition 4.8,

$$q \equiv q' + q'' \text{ with } \nu(p') \equiv_\alpha q' \text{ and } \nu(p'') \equiv_\alpha q''. \tag{6}$$

Suppose that

$(p' + p'', v) \xrightarrow{a(d)} (\overline{p}, v')$

and, by Definition 4.1, assume it is the case that

$$(p', v) \xrightarrow{a(d)} (\overline{p}, v')$$

Since $fv(p') \subseteq fv(p)$, and considering (5) and (6), we can apply induction hypothesis. So, there exists $(\overline{q}, \overline{v}')$ such that

$$(q', \overline{v}) \xrightarrow{a(d)} (\overline{q}, \overline{v}') \text{ and } \exists \nu'.\ \nu' \text{ is a renaming} \\ \wedge \nu'(fv(\overline{p})) = fv(\overline{q}) \wedge \nu'(\overline{p}) \equiv_\alpha \overline{q} \wedge v' \restriction fv(\overline{p}) = (\overline{v}' \circ \nu') \restriction fv(\overline{p}).$$

Now, by Definition 4.1,

$$(q' + q'', \overline{v}) \xrightarrow{a(d)} (\overline{q}, \overline{v}')$$

which prove this case.

The proof is analogous if we consider the subcase $(p'', v) \xrightarrow{a(d)} (\overline{p}, v')$.

Suppose $p \equiv \{\!|C|\!\}\ p'$. By the hypothesis and considering Definition 4.7, there exists a renaming ν such that $\nu(fv(\{\!|C|\!\}\ p')) = fv(q)$ and

$$(\exists f : C \to V.\ f \text{ is bijective } \wedge \nu(fv(p') \backslash C) \cap V = \emptyset \wedge \{\!|f(C)|\!\}\ \nu[f](p') \equiv_\alpha q) \quad (7)$$
$$\wedge\ v \restriction fv(\{\!|C|\!\}\ p') = (\overline{v} \circ \nu) \restriction fv(\{\!|C|\!\}\ p'). \quad (8)$$

So, by Definition 4.8,

$$q \equiv \{\!|C'|\!\}\ q' \text{ with} \\ C' \cap fv(\{\!|f(C)|\!\}\ \nu[f](p')) = \emptyset \\ \wedge (\exists g : f(C) \to C'.\ g \text{ is bijective } \wedge\ \iota[g] \circ \nu[f](p') \equiv_\alpha q') \quad (9)$$

If $x \in C$, then

$$v[C \hookleftarrow 0](x) = 0 = \overline{v}[C' \hookleftarrow 0](g \circ f(x)) = (\overline{v}[C' \hookleftarrow 0] \circ \iota[g] \circ \nu[f])(x)$$

Suppose now, $x \in fv(p) \backslash C$. Thus, considering (8), we have

$$v[C \hookleftarrow 0](x) = v(x) = \overline{v}(\nu(x)).$$

Because $x \in fv(p)$, $\nu(x) \in fv(q) = fv(\{\!|C'|\!\}\ q')$. So $\nu(x) \notin C'$. Thus,

$$\overline{v}(\nu(x)) = \overline{v}[C' \hookleftarrow 0](\nu(x)).$$

Now, since (7) and $x \notin C$,

$$\overline{v}[C' \hookleftarrow 0](\nu(x)) = (\overline{v}[C' \hookleftarrow 0](\nu[f](x))) = (\overline{v}[C' \hookleftarrow 0] \circ (\iota[g] \circ \nu[f]))(x).$$

So, we have that

$$v[C \hookleftarrow 0] \restriction (C \cup fv(p)) = (\overline{v}[C' \hookleftarrow 0] \circ (\iota[g] \circ \nu[f])) \restriction (C \cup fv(p)).$$

In particular, since $fv(p') \subseteq C \cup fv(p)$,

$$v[C \hookleftarrow 0] \restriction fv(p') = (\overline{v}[C' \hookleftarrow 0] \circ (\iota[g] \circ \nu[f])) \restriction fv(p'). \quad (10)$$

Suppose

$$(\{\!|C|\!\}\ p', v) \xrightarrow{a(d)} (p'', v').$$

By Definition 4.1,

$(p', v[C \leftarrow\!\!\shortmid 0]) \xrightarrow{a(d)} (p'', v')$.

By (9) and (10), we can apply the induction hypothesis. So, there exists $(q'', \overline{v}')$ such that

$(q', \overline{v}[C' \leftarrow\!\!\shortmid 0]) \xrightarrow{a(d)} (q'', \overline{v}')$ and $\exists \nu'.\ \nu'$ is a renaming
$\wedge\ \nu'(p'') \equiv_\alpha q'' \wedge v' \restriction fv(p'') = (\overline{v}' \circ \nu') \restriction fv(p'')$.

which implies, by Definition 4.1,

$(\{\!|C'|\!\}\ q', \overline{v}) \xrightarrow{a(d)} (q'', \overline{v}')$.

which prove this case.

Suppose $p \equiv \psi \mathrel{|\!\!>\!\!>} p'$. By the hypothesis and considering Definition 4.7, there exists a renaming ν such that $\nu(fv(\psi \mathrel{|\!\!>\!\!>} p')) = fv(q)$ and

$$\nu(\psi) \mathrel{|\!\!>\!\!>} \nu(p') \equiv_\alpha q \wedge v \restriction fv(\psi \mathrel{|\!\!>\!\!>} p') = (\overline{v} \circ \nu) \restriction fv(\psi \mathrel{|\!\!>\!\!>} p'). \tag{11}$$

So, by Definition 4.8,

$q \equiv \psi' \mathrel{|\!\!>\!\!>} q'$ with $\nu(\psi) = \psi' \wedge \nu(p') \equiv_\alpha q'$.

Suppose that

$(\psi \mathrel{|\!\!>\!\!>} p', v) \xrightarrow{a(d)} (p'', v')$.

By Definition 4.1,

$(p', v) \xrightarrow{a(d)} (p'', v')$ and $\models (v + d)(\psi)$.

Because $fv(p') \subseteq fv(p)$, by induction hypothesis, there exists $(q'', \overline{v}')$ such that

$$(q', \overline{v}) \xrightarrow{a(d)} (q'', \overline{v}') \text{ and } \exists \nu'.\ \nu' \text{ is a renaming} \wedge\ \nu'(fv(p'')) = fv(q'') \wedge \nu'(p'') \equiv_\alpha q'' \wedge v' \restriction fv(p'') = (\overline{v}' \circ \nu') \restriction fv(p''). \tag{12}$$

Considering (11), for all $d \in \mathbb{R}^{\geq 0}$,

$(v + d) \restriction fv(\psi \mathrel{|\!\!>\!\!>} p') = ((\overline{v} \circ \nu) + d) \restriction fv(\psi \mathrel{|\!\!>\!\!>} p') = ((\overline{v} + d) \circ \nu) \restriction fv(\psi \mathrel{|\!\!>\!\!>} p')$.

Because of that

$((\overline{v} + d) \circ \nu)(\psi) = (\overline{v} + d)(\nu(\psi)) = (\overline{v} + d)(\psi')$.

Thus,

$\models (\overline{v} + d)(\psi')$

So, together with (12), by Definition 4.1,

$(\psi' \mathrel{|\!\!>\!\!>} q', \overline{v}) \xrightarrow{a(d)} (q'', \overline{v}')$.

which proves this case. □

C Proof of Claim 6.9

Let $L_i = (S_i, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_0^i, \longrightarrow_i, \mathcal{U}^i)$, $i \in \{1,2\}$. So far, we introduce the notation $L_1 \underline{\leftrightarrow} L_2$ to mean that there is a timed bisimulation between the initial states of those TTS. Ambiguously, we will say that two states $s_1 \in S_1$ and $s_2 \in S_2$ are timed bisimilar (notation $s_1 \underline{\leftrightarrow} s_2$) if $(S_1, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_1, \longrightarrow_1, \mathcal{U}^1) \underline{\leftrightarrow} (S_2, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_2, \longrightarrow_2, \mathcal{U}^2)$. Now, we define:

Definition C.1 (Timed bisimulation up to $\underline{\leftrightarrow}$) Let $L_i = (S_i, \mathbf{A} \times \mathbb{R}^{\geq 0}, s_0^i, \longrightarrow_i, \mathcal{U}^i)$, $i \in \{1,2\}$, be two UTS. A *timed bisimulation up to* $\underline{\leftrightarrow}$ is a relation $R \subseteq S_1 \times S_2$ with $s_0^1 R s_0^2$ satisfying, for all $a(d) \in \mathbf{A} \times \mathbb{R}^{\geq 0}$, the following transfer properties:

1. if $s_1 R s_2$ and $s_1 \xrightarrow{a(d)}_1 s_1'$, then $\exists s_2' \in S_2 :\ s_2 \xrightarrow{a(d)}_2 s_2'$ and $s_1' \underline{\leftrightarrow} \overline{s}_0^1 R \overline{s}_0^2 \underline{\leftrightarrow} s_2'$ where $\overline{s}_0^i$ is the initial state of some TTS $\overline{L}_i$ ($i \in \{1,2\}$);

2. if $s_1 R s_2$ and $s_2 \xrightarrow{a(d)}_2 s_2'$, then $\exists s_1' \in S_1 :\ s_1 \xrightarrow{a(d)}_1 s_1'$ and $s_1' \underline{\leftrightarrow} \overline{s}_0^1 R \overline{s}_0^2 \underline{\leftrightarrow} s_2'$ where $\overline{s}_0^i$ is the initial state of some TTS $\overline{L}_i$ ($i \in \{1,2\}$); and

3. if $s_1 R s_2$, then $\mathcal{U}_d^1(s_1) \iff \mathcal{U}_d^2(s_2)$. □

It is not difficult to prove that if there is a timed bisimulation up to $\underline{\leftrightarrow}$ between two TTS, then they are timed bisimilar.

Claim 6.9. *Let $\overline{p}$ and $\overline{p}'$ be two terms in the extended language such that $\overline{p} \underline{\leftrightarrow} \overline{p}'$. Then, for all $v_0 \in \mathcal{C}$, there is a timed bisimulation R between $(\!|[\![\overline{p}]\!]^{\mathcal{T}}|\!)_{v_0}$ and $(\!|[\![\overline{p}']\!]^{\mathcal{T}}|\!)_{v_0}$. Define:*

$$
\begin{aligned}
S_1 \stackrel{\text{def}}{=}\ & \{((p||_A q, v), (p'||_A q, v'))|\ (p,v)\overline{R}_{var(q)}(p',v')\} \\
& \cup \{((\overline{\mathsf{ck}}(p)||_A q, v), (\overline{\mathsf{ck}}(p')||_A q, v'))|\ (p,\overline{v})\overline{R}_{var(q)}(p',\overline{v}') \\
& \quad \wedge\ v{\restriction}var(q) = v'{\restriction}var(q) \\
& \quad \wedge\ \exists d \in \mathbb{R}^{\geq 0}.\ (v{\restriction}var(p) = (\overline{v}[\kappa(p) \hookleftarrow 0] + d){\restriction}var(p) \\
& \qquad \wedge\ v'{\restriction}var(p') = (\overline{v}'[\kappa(p') \hookleftarrow 0] + d){\restriction}var(p'))\}
\end{aligned}
$$

$$
\begin{aligned}
S_1' \stackrel{\text{def}}{=}\ & \{((q||_A p, v), (q||_A p', v'))|\ (p,v)\overline{R}_{var(q)}(p',v')\} \\
& \cup \{((q||_A \overline{\mathsf{ck}}(p), v), (q||_A \overline{\mathsf{ck}}(p'), v'))|\ (p,\overline{v})\overline{R}_{var(q)}(p',\overline{v}') \\
& \quad \wedge\ v{\restriction}var(q) = v'{\restriction}var(q) \\
& \quad \wedge\ \exists d \in \mathbb{R}^{\geq 0}.\ (v{\restriction}var(p) = (\overline{v}[\kappa(p) \hookleftarrow 0] + d){\restriction}var(p) \\
& \qquad \wedge\ v'{\restriction}var(p') = (\overline{v}'[\kappa(p') \hookleftarrow 0] + d){\restriction}var(p'))\}
\end{aligned}
$$

$$
S_2 \stackrel{\text{def}}{=}\ \{((\overline{p} \between_A q, v_0), (\overline{p}' \between_A q, v_0))\} \cup S_1
$$

$$
S_2' \stackrel{\text{def}}{=}\ \{((q \between_A \overline{p}, v_0), (q \between_A \overline{p}', v_0))\} \cup S_1'
$$

$$S_3 \stackrel{\text{def}}{=} \{((\overline{p}|_A q, v_0), (\overline{p}'|_A q, v_0))\} \cup S_1$$

$$S_3' \stackrel{\text{def}}{=} \{((q|_A \overline{p}, v_0), (q|_A \overline{p}', v_0))\} \cup S_1'$$

All of those relations are timed bisimulation up to $\underline{\leftrightarrow}$.

Proof of the claim. First, we notice that it can be straightforwardly proven that, for any p and q, $(\overline{\mathsf{ck}}(p)||_A q, v) \underline{\leftrightarrow} (\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))||_A q, v)$. Moreover, we recall that the identity relation is a timed bisimulation, but in this case we are not going to make any explicit mention. We only prove the case of S_1, the rest can be proven similarly.

1. We prove the first transfer property.
Suppose $(p||_A q, v) \xrightarrow{a(d)} (r, \hat{v})$. Now, three subcases arises.

- $(p||_A q, v) \xrightarrow{a(d)} (p_0||_A \overline{\mathsf{ck}}(q), v[\kappa(p||_A q) \hookleftarrow 0] + d)$.

  By Definition 2.4,

  $$p||_A q \xrightarrow{a,\phi} p_0||_A \overline{\mathsf{ck}}(q) \text{ and } \models (v[\kappa(p||_A q) \hookleftarrow 0] + d)(\phi) \wedge \partial(p||_A q)).$$

  By rules in Table 8, $a \notin A$ and

  $$p \xrightarrow{a,\phi} p_0, \quad \kappa(p||_A q) = \kappa(p) \cup \kappa(q) \text{ and } \partial(p||_A q) = \partial(p) \wedge \partial(q).$$

  Since $ncv(p||_A q)$,

  $$\models (v[\kappa(p) \hookleftarrow 0] + d)(\phi) \text{ and } \models (v[\kappa(p) \hookleftarrow 0] + d)(\partial(p)).$$

  Now, by Definition 2.4,

  $$(p, v) \xrightarrow{a(d)} (p_0, v[\kappa(p) \hookleftarrow 0] + d).$$

  Since $(p, v) \overline{R}_{var(q)} (p', v')$,

  $$(p', v') \xrightarrow{a(d)} (p_0', v'[\kappa(p') \hookleftarrow 0] + d) \text{ and } (p_0, v[\kappa(p) \hookleftarrow 0] + d) \overline{R}_{var(q)} (p_0', v'[\kappa(p') \hookleftarrow 0] + d).$$

  By Definition 2.4,

  $$p' \xrightarrow{a,\phi'} p_0', \text{ and } \models (v'[\kappa(p') \hookleftarrow 0] + d)(\phi' \wedge \partial(p')).$$

  and by rules in Table 8,

  $$p'||_A q \xrightarrow{a,\phi'} p_0'||_A \overline{\mathsf{ck}}(q), \quad \kappa(p'||_A q) = \kappa(p') \cup \kappa(q) \text{ and } \partial(p'||_A q) = \partial(p') \wedge \partial(q).$$

  Moreover, since $ncv(p'||_A q)$,

  $$\models (v'[\kappa(p'||_A q) \hookleftarrow 0] + d)(\phi') \text{ and } \models (v'[\kappa(p'||_A q) \hookleftarrow 0] + d)(\partial(p'||_A q)).$$

So, by Definition 2.4,

$$(p'||_A q, v') \xrightarrow{a(d)} (p'_0||_A \overline{\mathsf{ck}}(q), v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Furthermore, $(p_0, v[\kappa(p)\leftarrow 0] + d)\overline{R}_{var(q)}(p'_0, v'[\kappa(p')\leftarrow 0] + d)$ implies

$$(v[\kappa(p)\leftarrow 0] + d)\restriction var(q) = (v'[\kappa(p')\leftarrow 0] + d)\restriction var(q)$$

and so

$$(v[\kappa(p||_A q)\leftarrow 0] + d)\restriction var(q) = (v'[\kappa(p'||_A q)\leftarrow 0] + d)\restriction var(q).$$

Since $fv(p_0) \cap bv(q) = fv(p'_0) \cap bv(q) = \emptyset$,

$$(v[\kappa(p||_A q)\leftarrow 0] + d)\restriction fv(p_0) = (v[\kappa(p)\leftarrow 0] + d)\restriction fv(p_0)$$
$$\text{and } (v[\kappa(p'||_A q)\leftarrow 0] + d)\restriction fv(p'_0) = (v[\kappa(p')\leftarrow 0] + d)\restriction fv(p'_0)$$

Now, because $var(q) = var(\overline{\mathsf{ck}}(q))$,

$$(p_0, v[\kappa(p||_A q)\leftarrow 0] + d)\overline{R}_{var(\overline{\mathsf{ck}}(q))}(p'_0, v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Thus,

$$(p_0||_A \overline{\mathsf{ck}}(q), v[\kappa(p||_A q)\leftarrow 0] + d)S_1(p'_0||_A \overline{\mathsf{ck}}(q), v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

- $(p||_A q, v) \xrightarrow{a(d)} (\overline{\mathsf{ck}}(p)||_A q_0, v[\kappa(p||_A q)\leftarrow 0] + d)$.
  By Definition 2.4,

  $$p||_A q \xrightarrow{a,\phi} \overline{\mathsf{ck}}(p)||_A q_0 \quad \text{and} \quad \models (v[\kappa(p||_A q)\leftarrow 0] + d)(\phi \wedge \partial(p||_A q)).$$

  By rules in Table 8, $a \notin A$ and

  $$q \xrightarrow{a,\phi} q_0, \quad \kappa(p||_A q) = \kappa(p) \cup \kappa(q) \text{ and } \partial(p||_A q) = \partial(p) \wedge \partial(q).$$

  Again, by rules in Table 8,

  $$p'||_A q \xrightarrow{a,\phi} \overline{\mathsf{ck}}(p')||_A q_0, \quad \kappa(p'||_A q) = \kappa(p') \cup \kappa(q) \text{ and } \partial(p'||_A q) = \partial(p') \wedge \partial(q).$$

  Since $(p, v)\overline{R}_{var(q)}(p', v')$, $\mathcal{U}_d(p, v)$ implies $\mathcal{U}_d(p', v')$, thus, by Definition 2.4,

  $$\models (v'[\kappa(p')\leftarrow 0] + d)(\partial(p')).$$

  Moreover, $v\restriction var(q) = v'\restriction var(q)$, and since $ncv(p||_A q)$ and $ncv(p'||_A q)$,

  $$\models (v'[\kappa(p'||_A q)\leftarrow 0] + d)(\phi \wedge \partial(p'||_A q)).$$

  So, by Definition 2.4,

  $$(p'||_A q, v') \xrightarrow{a(d)} (\overline{\mathsf{ck}}(p')||_A q_0, v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Furthermore, because $ncv(p||_A q)$, $ncv(p'||_A q)$ and $var(q_0) \subseteq var(q)$,

$$(v[\kappa(p||_A q) \leftarrow 0] + d) \restriction var(q_0) = (v'[\kappa(p'||_A q) \leftarrow 0] + d) \restriction var(q_0). \tag{13}$$

Now, suppose $x \in var(p)$, then $x \notin \kappa(q)$, which implies

$$(v[\kappa(p||_A q) \leftarrow 0] + d) \restriction var(p) = (v[\kappa(p) \leftarrow 0] + d) \restriction var(p). \tag{14}$$

Similarly,

$$(v'[\kappa(p'||_A q) \leftarrow 0] + d) \restriction var(p') = (v'[\kappa(p') \leftarrow 0] + d) \restriction var(p'). \tag{15}$$

Now, since $var(q_0) \subseteq var(q)$, $\overline{R}_{var(q)} \subseteq \overline{R}_{var(q_0)}$ which implies

$$(p, v)\overline{R}_{var(q_0)}(p', v'). \tag{16}$$

Finally, because of (13), (14), (15) and (16),

$$(\overline{\mathsf{ck}}(p)||_A q_0, v[\kappa(p||_A q) \leftarrow 0] + d) S_1 (\overline{\mathsf{ck}}(p')||_A q_0, v'[\kappa(p'||_A q) \leftarrow 0] + d).$$

- $(p||_A q, v) \xrightarrow{a(d)} (p_0||_A q_0, v[\kappa(p||_A q) \leftarrow 0] + d)$.

  By Definition 2.4,

  $$p||_A q \xrightarrow{a, \phi \wedge \phi''} p_0||_A q_0 \quad \text{and} \quad \models (v[\kappa(p||_A q) \leftarrow 0] + d)((\phi \wedge \phi'') \wedge \partial(p||_A q)).$$

  By rules in Table 8, $a \in A$ and

  $$p \xrightarrow{a, \phi} p_0, \quad q \xrightarrow{a, \phi''} q_0, \quad \kappa(p||_A q) = \kappa(p) \cup \kappa(q) \text{ and } \partial(p||_A q) = \partial(p) \wedge \partial(q).$$

  Since $ncv(p||_A q)$,

  $$\models (v[\kappa(p) \leftarrow 0] + d)(\phi \wedge \partial(p)).$$

  Now, by Definition 2.4,

  $$(p, v) \xrightarrow{a(d)} (p_0, v[\kappa(p) \leftarrow 0] + d).$$

  Since $(p, v)\overline{R}_{var(q)}(p', v')$,

  $$(p', v') \xrightarrow{a(d)} (p'_0, v'[\kappa(p') \leftarrow 0] + d) \text{ and } (p_0, v[\kappa(p) \leftarrow 0] + d)\overline{R}_{var(q)}(p'_0, v'[\kappa(p') \leftarrow 0] + d).$$

  By Definition 2.4,

  $$p' \xrightarrow{a, \phi'} p'_0, \text{ and } \models (v'[\kappa(p') \leftarrow 0] + d)(\phi' \wedge \partial(p')).$$

  and by rules in Table 8,

  $$p'||_A q \xrightarrow{a, \phi' \wedge \phi''} p'_0||_A q_0, \quad \kappa(p'||_A q) = \kappa(p') \cup \kappa(q) \text{ and } \partial(p'||_A q) = \partial(p') \wedge \partial(q).$$

Moreover, since $ncv(p||_A q)$ and $ncv(p'||_A q)$,

$$\models (v'[\kappa(p'||_A q)\leftarrow 0] + d)((\phi' \wedge \phi'') \wedge \partial(p'||_A q)).$$

So, by Definition 2.4,

$$(p'||_A q, v') \xrightarrow{a(d)} (p'_0||_A q_0, v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Furthermore, since $var(q_0) \subseteq var(q)$, $\overline{R}_{var(q)} \subseteq \overline{R}_{var(q_0)}$. Then

$$(p_0, v[\kappa(p)\leftarrow 0] + d)\overline{R}_{var(q_0)}(p'_0, v'[\kappa(p')\leftarrow 0] + d)$$

which implies

$$(v[\kappa(p)\leftarrow 0] + d)\restriction var(q_0) = (v'[\kappa(p')\leftarrow 0] + d)\restriction var(q_0)$$

and so

$$(v[\kappa(p||_A q)\leftarrow 0] + d)\restriction var(q_0) = (v'[\kappa(p'||_A q)\leftarrow 0] + d)\restriction var(q_0).$$

Since $fv(p_0) \cap bv(q) = fv(p'_0) \cap bv(q) = \emptyset$,

$$(v[\kappa(p||_A q)\leftarrow 0] + d)\restriction fv(p_0) = (v[\kappa(p)\leftarrow 0] + d)\restriction fv(p_0)$$
$$\text{and } (v[\kappa(p'||_A q)\leftarrow 0] + d)\restriction fv(p'_0) = (v[\kappa(p')\leftarrow 0] + d)\restriction fv(p'_0)$$

Now,

$$(p_0, v[\kappa(p||_A q)\leftarrow 0] + d)\overline{R}_{var(q_0)}(p'_0, v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Thus,

$$(p_0||_A q_0, v[\kappa(p||_A q)\leftarrow 0] + d)S_1(p'_0||_A q_0, v'[\kappa(p'||_A q)\leftarrow 0] + d).$$

Suppose $(\overline{\mathsf{ck}}(p)||_A q, v) \xrightarrow{a(d)} (r, \hat{v})$. Now, three subcases arises.

- $(\overline{\mathsf{ck}}(p)||_A q, v) \xrightarrow{a(d)} (p_0||_A \overline{\mathsf{ck}}(q), v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0] + d)$.
  By Definition 2.4,

  $$\overline{\mathsf{ck}}(p)||_A q \xrightarrow{a,\phi} p_0||_A \overline{\mathsf{ck}}(q) \text{ and } \models (v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0] + d)(\phi \wedge \partial(\overline{\mathsf{ck}}(p)||_A q)).$$

  By rules in Table 8, $a \notin A$ and

  $$p \xrightarrow{a,\phi} p_0, \quad \kappa(\overline{\mathsf{ck}}(p)||_A q) = \kappa(q) \text{ and } \partial(\overline{\mathsf{ck}}(p)||_A q) = \partial(p) \wedge \partial(q).$$

  By definition of S_1, there exists $\overline{v}$, $\overline{v}'$ and d' such that

  $$v\restriction var(p) = (\overline{v}[\kappa(p)\leftarrow 0] + d')\restriction var(p),$$
  $$v'\restriction var(p') = (\overline{v}'[\kappa(p')\leftarrow 0] + d')\restriction var(p') \text{ and } (p, \overline{v})\overline{R}_{var(q)}(p', \overline{v}'). \qquad (17)$$

Hence, since $ncv(\overline{\mathsf{ck}}(p)||_A q)$,

$$\models (\overline{v}[\kappa(p)\leftarrow\!\!\!\rightarrow 0] + d' + d)(\phi \wedge \partial(p)).$$

Thus, by Definition 2.4

$$(p, \overline{v}) \xrightarrow{a(d)} (p_0, \overline{v}[\kappa(p)\leftarrow\!\!\!\rightarrow 0] + d' + d).$$

which implies,

$$(p', \overline{v}') \xrightarrow{a(d)} (p'_0, \overline{v}'[\kappa(p')\leftarrow\!\!\!\rightarrow 0] + d' + d)$$
$$\text{and } (p_0, \overline{v}[\kappa(p)\leftarrow\!\!\!\rightarrow 0] + d' + d)\overline{R}_{var(q)}(p'_0, \overline{v}'[\kappa(p')\leftarrow\!\!\!\rightarrow 0] + d' + d). \qquad (18)$$

By Definition 2.4,

$$p' \xrightarrow{a,\phi'} p'_0 \text{ and } \models (\overline{v}'[\kappa(p')\leftarrow\!\!\!\rightarrow 0] + d' + d)(\phi' \wedge \partial(p')).$$

and by rules in Table 8,

$$\overline{\mathsf{ck}}(p')||_A q \xrightarrow{a,\phi'} p'_0||_A\overline{\mathsf{ck}}(q),\ \kappa(\overline{\mathsf{ck}}(p')||_A q) = \kappa(q) \text{ and } \partial(\overline{\mathsf{ck}}(p')||_A q) = \partial(p') \wedge \partial(q).$$

Moreover, since $ncv(\overline{\mathsf{ck}}(p')||_A q)$,

$$\models (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d)(\phi' \wedge \partial(\overline{\mathsf{ck}}(p')||_A q)).$$

So, by Definition 2.4,

$$(\overline{\mathsf{ck}}(p')||_A q, v') \xrightarrow{a(d)} (p'_0||_A\overline{\mathsf{ck}}(q), v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d).$$

Furthermore, by definition of S_1, $v\restriction var(q) = v'\restriction var(q)$, which implies

$$(v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow\!\!\!\rightarrow 0] + d)\restriction var(q) = (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d)\restriction var(q).$$

and because of (17), and since $ncv(\overline{\mathsf{ck}}(p')||_A q)$, $var(p_0) \subseteq var(p)$ and $var(p'_0) \subseteq var(p')$

$$(v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow\!\!\!\rightarrow 0] + d)\restriction var(p_0) = (\overline{v}[\kappa(p)\leftarrow\!\!\!\rightarrow 0] + d' + d)\restriction var(p_0)$$
$$\text{and } (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d)\restriction var(p'_0) = (\overline{v}'[\kappa(p')\leftarrow\!\!\!\rightarrow 0] + d' + d)\restriction var(p'_0).$$

Now, since $var(q) = var(\overline{\mathsf{ck}}(q))$, and considering (18),

$$(p_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow\!\!\!\rightarrow 0] + d)\overline{R}_{var(\overline{\mathsf{ck}}(q))}(p'_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d)$$

which implies

$$(p_0||_A\overline{\mathsf{ck}}(q), v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow\!\!\!\rightarrow 0] + d)S_1(p'_0||_A\overline{\mathsf{ck}}(q), v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow\!\!\!\rightarrow 0] + d).$$

- $(\overline{\mathsf{ck}}(p)||_A q, v) \xrightarrow{a(d)} (\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))||_A q_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d)$.

  By Definition 2.4,

  $$\overline{\mathsf{ck}}(p)||_A q \xrightarrow{a,\phi} \overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))||_A q_0, \text{ and } \models (v[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d)(\phi \wedge \partial(\overline{\mathsf{ck}}(p)||_A q)).$$

  By rules in Table 8, $a \notin A$ and

  $$q \xrightarrow{a,\phi} q_0, \quad \kappa(\overline{\mathsf{ck}}(p)||_A q) = \kappa(q) \text{ and } \partial(\overline{\mathsf{ck}}(p)||_A q) = \partial(p) \wedge \partial(q).$$

  Again, by rules in Table 8,

  $$\overline{\mathsf{ck}}(p')||_A q \xrightarrow{a,\phi} \overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p'))||_A q_0,$$
  $$\kappa(\overline{\mathsf{ck}}(p')||_A q) = \kappa(q) \text{ and } \partial(\overline{\mathsf{ck}}(p')||_A q) = \partial(p') \wedge \partial(q).$$

  Now, $(p, \overline{v})\overline{R}_{var(q)}(p', \overline{v}')$ for some $\overline{v}$, $\overline{v}'$ and $d' \in \mathbb{R}^{\geq 0}$ such that

  $$v \restriction var(p) = (\overline{v}[\kappa(p) \leftarrow 0] + d') \restriction var(p)$$
  $$\text{and } v' \restriction var(p') = (\overline{v}[\kappa(p') \leftarrow 0] + d') \restriction var(p'). \tag{19}$$

  Thus $\models (\overline{v}[\kappa(p) \leftarrow 0] + d' + d)(\partial(p))$, which implies $\mathcal{U}_{d'+d}(p, \overline{v})$ by Definition 2.4. So $\mathcal{U}_{d'+d}(p', \overline{v}')$ and, by Definition 2.4,

  $$\models (v'[\kappa(p') \leftarrow 0] + d' + d)(\partial(p')).$$

  Moreover, since $v \restriction var(q) = v' \restriction var(q)$, $ncv(\overline{\mathsf{ck}}(p)||_A q)$ and $ncv(\overline{\mathsf{ck}}(p')||_A q)$,

  $$\models (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d)(\phi \wedge \partial(\overline{\mathsf{ck}}(p')||_A q)).$$

  So, by Definition 2.4,

  $$(\overline{\mathsf{ck}}(p')||_A q, v') \xrightarrow{a(d)} (\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p'))||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d).$$

  We know that

  $$(\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))||_A q_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d) \underline{\leftrightarrow} (\overline{\mathsf{ck}}(p)||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d) \tag{20}$$

  and

  $$(\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p'))||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d) \underline{\leftrightarrow} (\overline{\mathsf{ck}}(p')||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d) \tag{21}$$

  Besides, since $var(q_0) \subseteq var(q)$, $(p, \overline{v})\overline{R}_{var(q)}(p', \overline{v}')$ implies

  $$(p, \overline{v})\overline{R}_{var(q_0)}(p', \overline{v}') \tag{22}$$

  and moreover

$$(v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0]+d)\restriction var(q_0) = (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow 0]+d)\restriction var(q_0). \quad (23)$$

Furthermore, because of (19) and the fact that $ncv(\overline{\mathsf{ck}}(p)||_A q)$ and $ncv(\overline{\mathsf{ck}}(p')||_A q)$,

$$(v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0]+d)\restriction var(p) = (\overline{v}[\kappa(p)\leftarrow 0]+d'+d)\restriction var(p)$$
$$\text{and } (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow 0]+d)\restriction var(p') = (\overline{v}[\kappa(p')\leftarrow 0]+d'+d)\restriction var(p'). \quad (24)$$

Finally, because of (22), (23) (24), (20) and (21),

$$(\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))||_A q_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0]+d) \underline{\leftrightarrow} S_1 \underline{\leftrightarrow}$$
$$(\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p'))||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q)\leftarrow 0]+d).$$

- $(\overline{\mathsf{ck}}(p)||_A q, v) \xrightarrow{a(d)} (p_0||_A q_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0]+d)$.

  By Definition 2.4,

  $$\overline{\mathsf{ck}}(p)||_A q \xrightarrow{a,\phi\wedge\phi''} p_0||_A q_0 \text{ and } \models (v[\kappa(\overline{\mathsf{ck}}(p)||_A q)\leftarrow 0]+d)((\phi\wedge\phi'')\wedge\partial(\overline{\mathsf{ck}}(p)||_A q)).$$

  By rules in Table 8, $a\in A$ and

  $$p \xrightarrow{a,\phi} p_0, \quad q \xrightarrow{a,\phi''} q_0, \quad \kappa(\overline{\mathsf{ck}}(p)||_A q) = \kappa(q) \text{ and } \partial(\overline{\mathsf{ck}}(p)||_A q) = \partial(p)\wedge\partial(q).$$

  By definition of S_1, there exists $\overline{v}$, $\overline{v}'$ and d' such that

  $$v\restriction var(p) = (\overline{v}[\kappa(p)\leftarrow 0]+d')\restriction var(p),$$
  $$v'\restriction var(p') = (\overline{v}'[\kappa(p')\leftarrow 0]+d')\restriction var(p') \text{ and } (p,\overline{v})\overline{R}_{var(q)}(p',\overline{v}'). \quad (25)$$

  Hence, since $ncv(\overline{\mathsf{ck}}(p)||_A q)$,

  $$\models (\overline{v}[\kappa(p)\leftarrow 0]+d'+d)(\phi\wedge\partial(p)).$$

  Thus, by Definition 2.4

  $$(p,\overline{v}) \xrightarrow{a(d)} (p_0, \overline{v}[\kappa(p)\leftarrow 0]+d'+d).$$

  which implies,

  $$(p',\overline{v}') \xrightarrow{a(d)} (p_0', \overline{v}'[\kappa(p')\leftarrow 0]+d'+d)$$
  $$\text{and } (p_0, \overline{v}[\kappa(p)\leftarrow 0]+d'+d)\overline{R}_{var(q)}(p_0', \overline{v}'[\kappa(p')\leftarrow 0]+d'+d).$$

  By Definition 2.4,

  $$p' \xrightarrow{a,\phi'} p_0' \text{ and } \models (\overline{v}'[\kappa(p')\leftarrow 0]+d'+d)(\phi'\wedge\partial(p')).$$

  and by rules in Table 8,

$\overline{\mathsf{ck}}(p')||_A q \xrightarrow{a,\phi'} p'_0||_A q_0$, $\kappa(\overline{\mathsf{ck}}(p')||_A q) = \kappa(q)$ and $\partial(\overline{\mathsf{ck}}(p')||_A q) = \partial(p') \wedge \partial(q)$.

Moreover, since $ncv(\overline{\mathsf{ck}}(p')||_A q)$ and (25),

$\models (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d)((\phi' \wedge \phi'') \wedge \partial(\overline{\mathsf{ck}}(p')||_A q))$.

So, by Definition 2.4,

$(\overline{\mathsf{ck}}(p')||_A q, v') \xrightarrow{a(d)} (p'_0||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d)$.

Moreover, since $var(q_0) \subseteq var(q)$, $\overline{R}_{var(q)} \subseteq \overline{R}_{var(q_0)}$. Then

$$(p_0, v[\kappa(p) \leftarrow 0] + d)\overline{R}_{var(q_0)}(p'_0, v'[\kappa(p') \leftarrow 0] + d) \tag{26}$$

By definition of S_1, $v \restriction var(q) = v' \restriction var(q)$, and since $var(q_0) \subseteq var(q)$,

$$(v[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d) \restriction var(q_0) = (v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d) \restriction var(q_0). \tag{27}$$

Furthermore, because of (25), and since $ncv(\overline{\mathsf{ck}}(p')||_A q)$, $var(p_0) \subseteq var(p)$ and $var(p'_0) \subseteq var(p')$

$$\begin{aligned}(v[\kappa(p||_A q) \leftarrow 0] + d) \restriction var(p_0) &= (v[\kappa(p) \leftarrow 0] + d' + d) \restriction var(p_0) \\ \text{and } (v[\kappa(p'||_A q) \leftarrow 0] + d) \restriction var(p'_0) &= (v[\kappa(p') \leftarrow 0] + d' + d) \restriction var(p'_0)\end{aligned} \tag{28}$$

Finally, because of (26), (26) and (28)

$(p_0||_A q_0, v[\kappa(\overline{\mathsf{ck}}(p)||_A q) \leftarrow 0] + d) S_1 (p'_0||_A q_0, v'[\kappa(\overline{\mathsf{ck}}(p')||_A q) \leftarrow 0] + d)$.

2. This transfer property is symmetric to the first one.

3. We omit this proof. It follows similar reasoning to the firs one. □