Abstraction in Real Time Process Algebra

A.S. Klusener

CWI

P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

e-mail: stevenk@cwi.nl

Abstract. In this paper we extend Real Time Process Algebra by the silent step τ. We start by giving the operational semantics and we find a characterizing law of which the soundness and the completeness is proven. By adding the integral construct we can interpret symbolic (untimed) process terms as timed processes. We investigate the resulting τ-equivalence and come to a delay bisimulation with a stronger root condition. Finally we test the applicability of this notion of real time abstraction by proving the PAR protocol (Positive Acknowledgement with Retransmission) correct.

1985 Mathematics Subject Classification: 68Q60.
1982 CR Categories: D.3.1, F.3.1, J.7.
Key Words & Phrases: Real Time, Process Algebra, ACP, Abstraction, Protocol Verification.
Note: This work is in part sponsored by ESPRIT Basic Research Action 3006, CONCUR. Some proofs have been omitted in this paper, they can be found in the full version which has appeared as CWI report CS9144 under the same title.

1 Introduction

In recent years much effort is paid to develop techniques for proving software systems correct w.r.t. to their specification. A motivation and an overview of these techniques can be found in [dR89]. In this paper we restrict ourselves to ACP ([BW90]), which is a Proces Algebra like CCS ([Mil89]) and CSP ([Hoa85]). The idea of protocol verification by using Process Algebra is that one has a specification and an implementation both formulated in the same language. One can abstract from the implementation details by renaming certain "internal" actions to the silent action τ. Then, one can apply the axioms of the algebra for proving the equality between the specification and the implementation. For examples of protocol verification in (untimed) Process Algebra we refer to [Bae90].

The most common notion of abstraction, weak bisimulation, is due to Milner [Mil80]. Van Glabbeek and Weijland introduced in [GW89] delay bisimulation and branching bisimulation, which are slightly different notions of abstraction.

In timed Process Algebra abstraction is not yet well developed. Only Wang studied abstraction in a timed Process Algebra (timed CCS) ([Wan90]), although his weak bisimulation is not a congruence. In another timed extension of a Process Algebra, Timed CSP ([RR88],[Ree89]), there is a special action $WAIT\ t$ which idles for t time units. In this

way an internal activity can be expressed. Similar constructs can be found in [MT90], [HR90]. It may be the case that the introduction of a τ action to Real Time ACP makes it more easy to compare Real Time ACP with other calculi.

In this paper a notion of abstraction in Real Time Process Algebra is proposed. As starting point we take the work of Baeten and Bergstra, they presented in [BB91] their Real Time Process Algebra BPA$\rho\delta$. In that paper they suggested already to interpret a timed τ as an explicit idling. We will investigate this idea more thoroughly.

We take BPA$\rho\delta$ (without integrals) and add the timed action $\tau(t)$. The operational semantics is given and we give a complete axiomatization. The addition of the integral construct allows us to interpret symbolic process terms as a special class of timed process terms. It comes out that the resulting subtheory can be considered as being a delay bisimulation with a strongly rooted condition. By generalizing the laws of earlier sections we obtain axioms for process terms with integrals. Finally, we show the use of this theory by giving a protocol verification which depends on time.

This paper is based on "absolute" time, thus the timestamps of the actions are interpreted from the starting point. This is not a serious point since all results can be formulated in "relative" time as well.

2 Adding the Silent Step to the Original Semantics

2.1 The Syntax

In this Section we give some intuition for timed processes by introducing the operational semantics of [BB91] for process expressions over Basic Real Time Process Algebra (BPA$\rho\delta$). We do not yet consider integration in this section. Let $A_{\delta\tau}$ be the set of actions, containing the constants δ (for inaction) and τ (for internal activity). The alphabet of the theory BPA$\rho\delta\tau$ is

$$A_{\delta\tau}^{time} = \{a(t) | a \in A_{\delta\tau}, t \in \mathbb{R}^{\geq 0}\}$$

Similarly we use A_{τ}^{time}, as the set of timed actions without timed δ's. In the sequel we refer to actions from $A_{\delta\tau}$ as symbolic actions and we refer to actions from $A_{\delta\tau}^{time}$ as timed actions. Moreover, process expressions are simply called terms. The set $\mathcal{T}$ of (closed) terms over BPA$\rho\delta\tau$ is generated by the alphabet A_{δ}^{time} and the binary operators $+$ for alternative composition and $\cdot$ for sequential composition and the operator $\gg$, called the *(absolute) time shift.*

The *(absolute) time shift*, $\gg$, takes a nonnegative real number and a process term; $t \gg X$ denotes that part of X which starts after t. The set $\mathcal{T}$ with typical elements p, p_1, p_2 is defined in the following way, where $a \in A_{\delta\tau}$ and $r \in \mathbb{R}^{\geq 0}$:

$$p \in \mathcal{T} \quad p := a(r) \mid p_1 + p_2 \mid p_1 \cdot p_2 \mid r \gg p$$

Syntactical equivalence is denoted by $\equiv$, syntactical equivalence modulo associativity and commutativity of the $+$ is denoted by $\simeq$. Equivalence within a theory $\mathcal{TH}$ is denoted by $\mathcal{TH} \vdash p = q$ or simply $p = q$ when the theory is clear from the context. $\delta(0)$ is abbreviated by δ.

There are three functions defined by induction. $U(p)$ is the *ultimate delay* of p and $S(p)$ is the *earliest start time* of p and $L(p)$ is the *latest start time.* We need an auxiliary

function *inittime*; $inittime(p)$ contains all time stamps at which p can perform an initial action

These functions already occur in [Klu91]. a is taken from A_τ

$$\begin{array}{llll} U(a(t)) & = & t & inittime(a(t)) = \{t\} \\ U(\delta(t)) & = & t & inittime(\delta(t)) = \emptyset \\ U(p \cdot q) & = & U(p) & inittime(p \cdot q) = inittime(p) \\ U(p+q) & = & max(U(p), U(q)) & inittime(p+q) = inittime(p) \cup inittime(q) \\ U(r \gg p) & = & max(r, U(p)) & inittime(r \gg p) = inittime(p) \cap < r, \omega > \end{array}$$

We take $max(\emptyset) = min(\emptyset) = 0$ and we define $S(p) = min(inittime(p))$ and $L(p) = max(inittime(p))$.

2.2 The Original Semantics

The semantics of Baeten and Bergstra ([BB91]) assigns to every term (in $\mathcal{T}$) a transition system in which each state is a pair consisting of a term and a point in time and in which each transition is labeled by a timed (non δ) action.

For example the process $a(1)$ starts in state $< a(1), 0 >$, denoting that each process starts at 0. From $< a(1), 0 >$ an *idle* transition is possible to a state of the form $< a(1), t >$ with $0 < t < 1$. An *idle* transition is a transition in which only the time component is increased without executing any action. In general, from each state $< a(1), t >$ an *idle* transition is possible to $< a(1), t' >$, whenever $t < t' < 1$. Furthermore, from each state $< a(1), t >$ a terminating $a(1)$-transition to $< \surd, 1 >$ is possible whenever $t < 1$.

For technical reasons we add in this paper a boolean value to each state, which is initialized on F. So, the process $a(1)$ starts in $< a(1), 0, F >$. An *idle* transition does not change the boolean value. As soon as an action is executed the value is set to T. Once the value is set to T it remains T throughout the execution of the proces.

For the moment it suffices to say that we need the boolean value to distinguish root states from internal states. A root state is a state with time 0 or a state which can be reached from a state with time 0 by idling only.

Within this semantics the transition system concerns three relations

$$\begin{array}{lllllll} Step & \subseteq & (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{T,F\}) & \times & A^{time} & \times & (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{T,F\}) \\ Idle & \subseteq & (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{T,F\}) & \times & (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{T,F\}) & & \\ Terminate & \subseteq & (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{T,F\}) & \times & A^{time} & \times & \mathbb{R}^{\geq 0} \end{array}$$

These three relations are defined as the least relations satisfying the action rules given in Table 1. The definition of an operational semantics by giving action rules is due to Plotkin ([Plo81]). We write

$$\begin{array}{lll} < x,t,b > \xrightarrow{a(r)} < x',t',b' > & \text{for} & (< x,t,b >, a(r), < x',t',b' >) \in Step \\ < x,t,b > \longrightarrow < x',t',b' > & \text{for} & (< x,t,b >, < x',t',b' >) \in Idle \\ < x,t,b > \xrightarrow{a(r)} < \surd, t' > & \text{for} & (< x,t,b >, a(r), t') \in Terminate \end{array}$$

The transition relation will be defined such that it is guaranteed that

$$\begin{array}{lll} < x,t,b > \xrightarrow{a(r)} < x',t',b' > & \Longrightarrow & t < r \wedge t' = r \wedge b' = T \\ < x,t,b > \longrightarrow < x',t',b' > & \Longrightarrow & t < t' \wedge b = b' \\ < x,t,b > \xrightarrow{a(r)} < \surd, t' > & \Longrightarrow & t < r \wedge t' = r \end{array}$$

As notion of equivalence we have strong bisimulation; every *step* or *idle* transition on the left hand side has to be matched with an associated *step* or *idle* transition on the right hand side. A bisimulation relation on $(\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{F,T\}) \times (\mathcal{T} \times \mathbb{R}^{\geq 0} \times \{F,T\})$ is defined in the obvious way. Two terms p, q are bisimilar, denoted by $p \leftrightarrow_{orig} q$, if there is a bisimulation relation containing $(< p, 0, F >, < q, 0, F >)$. The action rules are given in Table 1.

From the atomic rules for $\tau(r)$ we see that executing the silent step τ is modeled by an *idle* step which changes the process term in the state. Therefore we have to cover now as well cases where idling may change the process term in the state.

2.3 Some Process Diagrams

The transition system of the term $a(1)$ can be represented by the left-hand process diagram given in Figure 1. A process diagram is simply a pictorial representation of a transition system. It is not possible to make a picture of the transition system itself, since it has uncountably many transitions. The intuition behind such a process diagram is that the process starts in the top-point. It can idle by going to a lower point without crossing any line, whereas the execution of an action a at time r is reflected by going to a dashed line at level r labeled with a. Only dashed lines may be crossed, after landing on them.

A very particular set of atomic actions is the set of $\delta(r)$-terms. $\delta(1)$ can do nothing more then idling until 1. Thus the root node is $< \delta(1), 0 >$ and from each state $< \delta(1), t >$ an *idle* transition to $< \delta(1), t' >$ is possible, whenever $t < t' < 1$.

The transition system of $p + q$ is defined in terms of the transition systems of p and q. The behaviour of $p + q$ can be considered as the "union" of the behaviour of p and that of q.

A state μ (in Figure 1) is of the form $< a(1) + b(2), t >$ with $0 < t < 1$. From μ both a terminating $a(1)$-transition to $< \surd, 1 >$ and a terminating $b(2)$-transition to $< \surd, 2 >$ are possible. However, from a state like ν of the form $< a(1) + b(2), t >$ with $1 \leq t < 2$ only a terminating $b(2)$-transition to $< \surd, 2 >$ is possible. Hence, by idling from $< a(1) + b(2), t_0 >$ to $< a(1) + b(2), t_1 >$ with $0 < t_0 < 1 \leq t_1 < 2$ we have lost the option of executing the $a(1)$-summand. Thus one could say that a choice has been made at time 1; after the choice has been made for $b(2)$ the summand $a(1)$ has become redundant.

The transition system of $a(1) + \delta(1)$ consists of exactly the same relations as the transition system of $a(1)$. The summand $\delta(1)$ contributes only *idle* steps which are con-

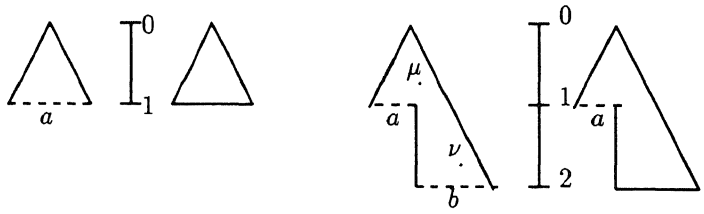


Figure 1: Process Diagrams of the terms $a(1)$,$\delta(1)$, $a(1) + b(2)$ and $a(1) + \delta(2)$

$$\frac{t < s < r}{< a(r), t, b > \longrightarrow < a(r), s, b >} \qquad \frac{t < r}{< a(r), t, b > \xrightarrow{a(r)} < \surd, r >}$$

$$\frac{t < s < r}{< \tau(r), t, b > \longrightarrow < \tau(r), s, b >} \qquad \frac{t < r}{< \tau(r), t, b > \longrightarrow < \surd, r >}$$

$$\frac{t < s < r}{< \delta(r), t, b > \longrightarrow < \delta(r), s, b >}$$

$$\frac{< p, t, F > \longrightarrow < p, r, F >}{< p + q, t, b > \longrightarrow < p + q, r, b >}$$

$$\frac{< p, t, b > \xrightarrow{a(r)} < \surd, r >}{< p + q, t, b > \xrightarrow{a(r)} < \surd, r >} \qquad \frac{< p, t, b > \xrightarrow{a(r)} < p', r, b' >}{< p + q, t, b > \xrightarrow{a(r)} < p', r, b' >}$$

$$\frac{< p, t, b > \longrightarrow < \surd, r >}{< p + q, t, b > \longrightarrow < \surd, r >} \qquad \frac{< p, t, F > \longrightarrow < p', r, T >}{< p + q, t, b > \longrightarrow < p', r, T >}$$

And similar rules for $q + p$

$$\frac{< p, t, b > \xrightarrow{a(r)} < \surd, r >}{< p \cdot q, t, b > \xrightarrow{a(r)} < q, r, T >} \qquad \frac{< p, t, b > \xrightarrow{a(r)} < p', r, b' >}{< p \cdot q, t, b > \xrightarrow{a(r)} < p' \cdot q, r, b' >}$$

$$\frac{< p, t, b > \longrightarrow < \surd, r >}{< p \cdot q, t, b > \longrightarrow < q, r, T >} \qquad \frac{< p, t, b > \longrightarrow < p', r, b' >}{< p \cdot q, t, b > \longrightarrow < p' \cdot q, r, b' >}$$

$$\frac{t < r < s}{< s \gg p, t, b > \longrightarrow < s \gg p, r, b >}$$

$$\frac{r > s \quad < p, t, b > \xrightarrow{a(r)} < \surd, r >}{< s \gg p, t, b > \xrightarrow{a(r)} < \surd, r >} \qquad \frac{r > s \quad < p, t, b > \xrightarrow{a(r)} < p', r, b' >}{< s \gg p, t, b > \xrightarrow{a(r)} < p', r, b' >}$$

$$\frac{r > s \quad < p, t, b > \longrightarrow < \surd, r >}{< s \gg p, t, b > \longrightarrow < \surd, r >} \qquad \frac{r > s \quad < p, t, b > \longrightarrow < p', r, b' >}{< s \gg p, t, b > \longrightarrow < p', r, b' >}$$

Table 1: The Original Transition System of BPA$\rho\delta\tau$, ($a \in A$, $r, s > 0$)

tributed by the summand $a(1)$ as well, hence we may consider the $\delta(1)$ summand as being redundant.

However if we consider $a(1)+\delta(2)$, the $\delta(2)$ summand contributes *idle* transitions which are not contributed by $a(1)$, since $\delta(2)$ has *idle* transitions to points in time between 1 and 2. The transition system of $a(1) + \delta(2)$ can be represented by the process diagram on the right-hand side in Figure 1.

The last operator we introduce is the *(absolute) time shift* denoted by $\gg$, which takes a real number s and a process X and delivers that part of X which starts after s. Hence, before s it can only *idle* or do a transition to a state after s.

2.4 The Closure Rules

In the original transition system of Baeten and Bergstra, thus the one without silent actions and without a boolean value in the state, the following property was guaranteed:

$$< p,t > \longrightarrow < p,s > \ \wedge \ < p,s > \xrightarrow{a(r)} < p',r > \quad \Longrightarrow \quad < p,t > \xrightarrow{a(r)} < p',r >$$

Since we require this property also in the context with silent steps (where idling may change the process term), we need the following *closure* rules. In the sequel we will discuss why these closure rules may only be applied on internal states.

$$\frac{< p,t,T > \longrightarrow < p',t',T > \quad < p',t',T > \xrightarrow{a(r)} < \surd,r >}{< p,t,T > \xrightarrow{a(r)} < \surd,r >} \qquad \frac{< p,t,T > \longrightarrow < p',t',T > \quad < p',t',T > \longrightarrow < \surd,r >}{< p,t,T > \longrightarrow < \surd,r >}$$

$$\frac{< p,t,T > \longrightarrow < p',t',T > \quad < p',t',T > \xrightarrow{a(r)} < p'',r,T >}{< p,t,T > \xrightarrow{a(r)} < p'',r,T >} \qquad \frac{< p,t,T > \longrightarrow < p',t',T > \quad < p',t',T > \longrightarrow < p'',r,T >}{< p,t,T > \longrightarrow < p'',r,T >}$$

Table 2: The Closure Rules, ($a \in A$, $r,s > 0$)

In Figure 2 we see process diagrams corresponding to the terms $a(1)\cdot(\tau(2)\cdot b(3)+c(3))$ and $a(1)\cdot(b(3)+\tau(2)\cdot c(3))$. On the left hand side the process diagrams without closure rules are given; without closure rules the terms are certainly not bisimilar. On the right hand side the process diagrams with closure rules are given; the two terms have become bisimilar.

Since in both terms the two summands are in a context $a(1)\cdot(...)$ we know that there are no other summands involved, hence the $\tau(2)$ action determines a moment of choice, namely at time 2. But it is not relevant whether the $\tau(2)$ is on the left hand side of the $+$ or on the right hand side. Consider $\tau(2)\cdot b(3) + c(3)$, thus without context $a(1)\cdot(...)$, then we can not say that the choice for the $c(3)$ is made at 2. This becomes clear when we would add $d(3)$. In the next Section we discuss this in more detail.

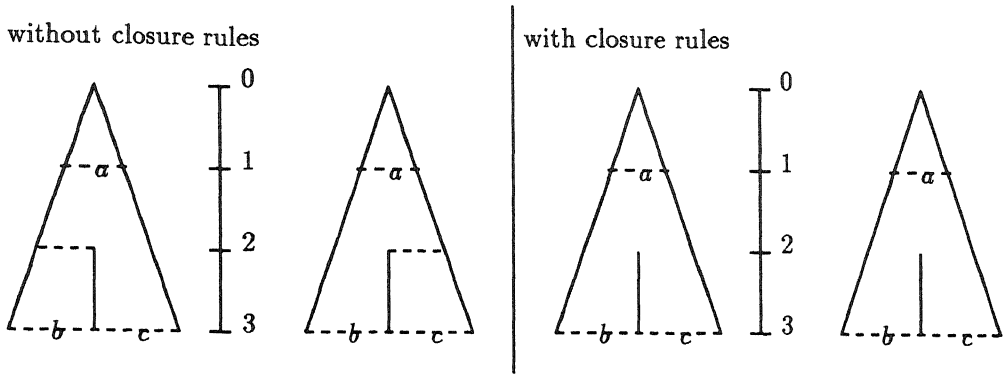


Figure 2: Process Diagrams for $a(1) \cdot (\tau(2) \cdot b(3) + c(3))$ and $a(1) \cdot (b(3) + \tau(2) \cdot c(3))$

In the sequel we refer to *idle* steps which are generated by the closure rules as *implicit (idle) steps.* Similarly we have *explicit (idle) steps.*

2.5 Closure Rules and Internal States

Now we can discuss the need for distinguishing root states from internal states. Assume that we would not make this distinction, so forget about the boolean values. And assume that we would apply the closure rules on the root level as well. Then we would have

$$a(1) + \tau(1) \cdot b(2) \quad \leftrightarrows \quad a(1) + b(2)$$

But if we add $c(2)$ then

$$a(1) + \tau(1) \cdot b(2) + c(2) \quad \not\leftrightarrows \quad a(1) + b(2) + c(2)$$

Since in the left hand process the choice for the $b(2)$-summand is made at time 1, while in the right hand side the corresponding choice is made at time 2. Thus in case of $a(1)+b(2)$ it is not right to say that the choice for the $b(2)$ is made at 1, since other summands, such as $c(2)$, may be there in the context.

If we put both terms in a sequential composition after $d(1)$ then

$$d(1) \cdot (a(1) + \tau(1) \cdot b(2)) \quad \not\leftrightarrows \quad d(1) \cdot (a(1) + b(2))$$

because only the right hand process has an option of doing the b at time 2.

So, when the closure rules are applied from the root level as well bisimulation equivalence is not a congruence. By making the distinction between root states and internal states we can apply the closure rules only for internal states and bisimulation equivalence is a congruence. For example:

$$a(1) + \tau(1) \cdot b(2) \quad \not\leftrightarrows \quad a(1) + b(2)$$

3 An Alternative Operational Semantics

3.1 Encoding the Course of Time in the Process Terms

In the previous Section the operational semantics is presented according to [BB91]. Each state consisted of three components. In this Section we give a transition relation, in which the course of time is encoded in the prefix by an occurrence of the $\gg$ operator. In [Klu91] a similar operational semantics is given. Since no abstraction was considered in that paper it was not necessary to model the *idle* steps there. Hence, the transition relation became finite for (recursion free) terms without integration. Now, we encorporate the silent step into this semantics; *idle* steps which determine a moment of choice are modeled by τ-steps. Moreover, we define a notion of equivalence, called *timed weak* bisimulation, which coincides with $\underline{\leftrightarrow}_{orig}$.

In the semantics for ACP, as presented in [Gla87], we have the following transitions

$$a \xrightarrow{a} \surd \quad \text{and} \quad a \cdot p \xrightarrow{a} p$$

In a real time setting we have to take the time stamps into account. In $a(r) \cdot p$, after doing the $a(r)$ action, only that part of p can be done which starts after r, which is denoted by $r \gg p$. Hence, after the addition of time we have the following transitions (we have also added the boolean values).

$$< a(r), b > \xrightarrow{a(r)} \surd \quad \text{and} \quad < a(r) \cdot p, b > \xrightarrow{a(r)} < r \gg p, T >$$

If p can perform an action $b(t)$ then $s \gg p$ can perform this action only if $t > s$.

3.2 The Transition System Specification

In table 3 an alternative operational semantics is given. Only the *implicit* τ-rule is new here, the other ones are taken from [Klu91]. This *implicit* τ-rule models a moment of choice by a τ-step. The alternative operational semantics given in this Section concerns two relations:

$$\begin{array}{lllllll} Step & \subseteq & (\mathcal{T} \times \{F,T\}) & \times & A_{\delta\tau}^{time} & \times & (\mathcal{T} \times \{F,T\}) \\ Terminate & \subseteq & (\mathcal{T} \times \{F,T\}) & \times & A_{\delta\tau}^{time} & & \end{array}$$

These relations are defined as the least relations satisfying the action rules of Table 3. We write:

$$\begin{array}{ll} < p, b > \xrightarrow{a(r)} < p', b' > & \text{for } ((p,b), a(r), (p',b')) \in Step \\ < p, b > \xrightarrow{a(r)} \surd & \text{for } ((p,b), a(r)) \in Terminate \end{array}$$

Again it is guaranteed that $b' = T$ if $< p, b > \xrightarrow{a(r)} < p', b' >$. This can be shown by induction on the length of the derivation. It is also guaranteed that

$$a \in A_\tau \; : < p, T > \xrightarrow{a(r)} < p', T > \quad \Longrightarrow \quad r = S(p)$$

All initial actions of p with a time stamp greater than $S(p)$ are postponed till a later state, as is shown in Figure 3. Two terms p and q are bisimilar, denoted by $p \underline{\leftrightarrow}_{tw} q$, if there

$$< a(r), b > \xrightarrow{a(r)} \surd$$

$$\frac{< p, F > \xrightarrow{a(r)} \surd}{< p+q, F > \xrightarrow{a(r)} \surd} \qquad \frac{< p, F > \xrightarrow{a(r)} < p', T >}{< p+q, F > \xrightarrow{a(r)} < p', T >}$$

$$\frac{r \leq S(q) \quad < p, T > \xrightarrow{a(r)} \surd}{< p+q, T > \xrightarrow{a(r)} \surd} \qquad \frac{r \leq S(q) \quad < p, T > \xrightarrow{a(r)} < p', T >}{< p+q, T > \xrightarrow{a(r)} < p', T >}$$

And similar rules for $q + p$

$$\frac{< p, b > \xrightarrow{a(r)} \surd}{< p \cdot q, b > \xrightarrow{a(r)} < r \gg q, T >} \qquad \frac{< p, b > \xrightarrow{a(r)} < p', T >}{< p \cdot q, b > \xrightarrow{a(r)} < p' \cdot q, T >}$$

$$\frac{s < r \quad < p, b > \xrightarrow{a(r)} \surd}{< s \gg p, b > \xrightarrow{a(r)} \surd} \qquad \frac{s < r \quad < p, b > \xrightarrow{a(r)} < p', T >}{< s \gg p, b > \xrightarrow{a(r)} < p', T >}$$

δ-rule

$$\frac{U(p) > L(p)}{< p, b > \xrightarrow{\delta(U(p))} \surd}$$

implicit τ-rule

$$\frac{s = S(r \gg p) < U(s \gg p)}{< r \gg p, T > \xrightarrow{\tau(s)} < s \gg p, T >}$$

Table 3: An Alternative Operational Semantics, ($a \in A_\tau$, $r, s > 0$)

is a *timed weak* bisimulation relation containing $(< p, F >, < q, F >)$. In the following definition $< p, T > \stackrel{a(r)}{\Longrightarrow} < p', T >$ abbreviates $< p, T > \xrightarrow{\tau(t_1)} < p_1, T > ... < p_k, T > \xrightarrow{a(r)} < p', T >$ for some $k \geq 0$, moreover a is take from $A_{\delta\tau}$ and a' is taken from A_δ.

Definition 3.2.1 *$R \subset (\mathcal{T} \times \{T, F\}) \times (\mathcal{T} \times \{T, F\})$ is a timed weak bisimulation relation iff it is symmetric and $(< p, b >, < q, b >) \in R$ implies that*

- *if $b = F$ and $< p, F > \xrightarrow{a(r)} < p', T >$ then there is a q' such that $< q, F > \xrightarrow{a(r)} < q', T >$ and $(< p', T >, < q', T >) \in R$.*
- *if $b = F$ and $< p, F > \xrightarrow{a(r)} \surd$ then $< q, F > \xrightarrow{a(r)} \surd$.*
- *if $b = T$ and $< p, T > \xrightarrow{a'(r)} < p', T >$ then there is a q' such that $< q, T > \stackrel{a'(r)}{\Longrightarrow} < q', T >$ and $(< p', T >, < q', T >) \in R$.*

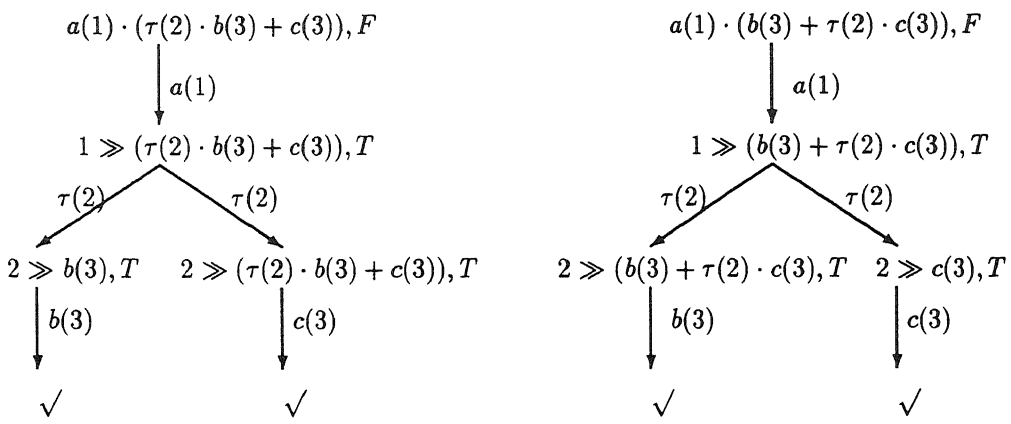


Figure 3: Transition Systems for $a(1)\cdot(\tau(2)\cdot b(3)+c(3))$ and $a(1)\cdot(b(3)+\tau(2)\cdot c(3))$

- *if* $b=T$ *and* $<p,T>\xrightarrow{\tau(r)}<p',T>$ *then either* $(<p',T>,<q,T>)\in R$ *or there is a* q' *such that* $<q,T>\stackrel{\tau(r)}{\Longrightarrow}<q',T>$ *and* $(<p',T>,<q',T>)\in R$
- *if* $b=T$ *and* $<p,T>\xrightarrow{a(\tau)}\surd$ *then* $<q,T>\stackrel{a(\tau)}{\Longrightarrow}\surd$.

The δ-rule generates transitions labeled with $\delta(t)$, for example if $t<2$ then

$$<t\gg(a(1)+\delta(2)),b>\xrightarrow{\delta(2)}\surd$$

These deadlock transitions are needed to distinguish $a(1)+\delta(2)$ from $a(1)+\delta(3)$. Note that $<a(1)+\delta(1),b>$ does not have a *deadlock* transition at all. *Deadlock* transitions are discussed exhaustively in [Klu91], but they are not important for discussing the silent step. Hence, they will not be mentioned anymore in this paper.

In this transition relation we have *implicit* τ-steps only when they determine a moment of choice. Consider the state $<r\gg p,T>$. Assume that the first moment in time at which $r\gg p$ can do an action is s (e.g. $S(r\gg p)=s$). Furthermore, assume that it can idle till a moment after s (e.g. $U(r\gg p)>s$). Then we say that s is a moment of choice; either the actions with time stamp s are executed or the idling continues and the actions with timestamp s are dropped from the computation. Since there are only finitely many of those moments of choice we have only finitely many τ-steps as well. In Figure 3 the states $<2\gg b(3),T>$ and $<2\gg(b(3)+\tau(2)\cdot c(3),T>$ may be related by a *timed weak* bisimulation relation. In Section 5 we will give a corresponding semantics where we can use the standard notion of strong bisimulation. The price to pay, however, is that we have to allow infinite many *implicit* τ steps there.

We can prove similarly as in [Klu91]:

Lemma 3.2.2 *Bisimulation equivalence in the alternative operational semantics equals bisimulation equivalence in the original operational semantics.*

$$p\leftrightarrows_{tw}q\quad\Longleftrightarrow\quad p\leftrightarrows_{orig}q$$

Thus we may omit the subscripts of $\leftrightarrows_{tw}$ and $\leftrightarrows_{orig}$. In Section 5 we will prove that bisimulation equivalence is a congruence.

4 The Theory BPA$\rho\delta\tau$

4.1 A Characterizing Law

From the operational semantics we know that a τ-action may be removed if it does not determine a choice. Moreover, this τ-removal can only be applied 'within' a term and not at the level of the root. This is exactly characterized by the τ-law given in Table 4.1. The theory BPA$\rho\delta$ can be found in [BB91] and [Klu91]

TAU1	$t < r < U(X) \ \wedge \ U(Y) \leq r$
$a(t) \cdot (\tau(r) \cdot X + Y) = a(t) \cdot (r \gg X + Y)$	

Table 4: BPA$\rho\delta\tau$ = BPA$\rho\delta$ + TAU1

Lemma 4.1.1 *The axiom TAU1 is sound w.r.t. to bisimulation equivalence.*

Theorem 4.1.2 *Soundness of BPA$\rho\delta\tau$*

$$BPA\rho\delta\tau \vdash p = q \quad \Longrightarrow p \leftrightarrows q$$

Proof. A theory is sound w.r.t. to an equivalence if all the axioms are sound and if the equivalence is a congruence w.r.t. to all operators. Since the bisimulation equivalence introduced in this paper identifies more than the one of [Klu91] we may conclude that all the axioms of BPA$\rho\delta$ are still sound. Moreover, in Lemma 4.1.1 we have stated the soundness of the additional τ-law and we will show that bisimulation equivalence is a congruence in Section 5. □

Lemma 4.1.3 *Assume $t < r < min(S(X), S(Y))$ and $U(Z) \leq r$ then*

τ-removal	$a(t) \cdot \tau(r) \cdot X$	$=$	$a(t) \cdot X$
τ-swap	$a(t) \cdot (\tau(r) \cdot X + Y)$	$=$	$a(t) \cdot (X + \tau(r) \cdot Y)$
τ-swap	$a(t) \cdot (\tau(r) \cdot X + Y + Z)$	$=$	$a(t) \cdot (X + \tau(r) \cdot Y + Z)$

Proof.

$$\begin{aligned} a(t) \cdot \tau(r) \cdot X &= a(t) \cdot (\tau(r) \cdot X + \delta) \\ &= a(t) \cdot (X + \delta) \\ &= a(t) \cdot X \end{aligned}$$

$$\begin{aligned} a(t) \cdot (\tau(r) \cdot X + Y) &= a(t) \cdot (\tau(r) \cdot X + \tau(r) \cdot Y) \\ &= a(t) \cdot (X + \tau(r) \cdot Y) \end{aligned}$$

$$\begin{aligned} a(t) \cdot (\tau(r) \cdot X + Y + Z) &= a(t) \cdot (\tau(r) \cdot X + \tau(r) \cdot Y + Z) \\ &= a(t) \cdot (X + \tau(r) \cdot Y + Z) \end{aligned}$$

□

4.2 Completeness

In this Section we prove the completeness of BPA$\rho\delta\tau$, e.g. we have to prove that if two terms are bisimilar then there is a derivation in BPA$\rho\delta\tau$ which proves them equal. We construct for each pair of bisimilar terms another pair of terms by adding τ-actions, such that the resulting pair is also bisimilar in the semantics without closure rules. We only add τ-actions in one term if there is already an associated τ in the other term. Or, put in other words, if implicit idling is matched with explicit idling then the implicit idling is rewritten into an explicit idling. Thus the pair

$$\begin{array}{lll} (& a(1)\cdot b(3)\cdot d(4) & , \quad a(1)\cdot\tau(2)\cdot b(3)\cdot(d(4)+d(4)) \quad) \\ \text{is rewritten into} & & \\ (& a(1)\cdot\tau(2)\cdot b(3)\cdot d(4) & , \quad a(1)\cdot\tau(2)\cdot b(3)\cdot(d(4)+d(4)) \quad) \end{array}$$

Since we are working in absolute time and since we have timed δ's, we allow terms with a lot of "junk" (redundant parts) in it. A *basic* term is a term without "junk". If we consider a term like $a(2)\cdot b(1)$, then the b can never be executed at 1 after we have executed the a at 2. Thus a *deadlock* will be encountered after executing the a at 2. Hence, we can rewrite the term $a(2)\cdot b(1)$ into the *basic* term $a(2)\cdot\delta$, where the *deadlock* appears explicitly. Similarly we can remove all *redundant* δ's, for example the $\delta(2)$ is *redundant* in the term $a(2)+\delta(2)$.

For the formal definition of *basic* terms and for further details we refer to [Klu91]. The set of *basic* terms is denoted by $\mathcal{B}$. Next, we rewrite all *basic* terms of the form

$$\begin{array}{ll} \sum_i a_i(r)\cdot p_i + \sum_j b_j(r) + q & \quad \textit{with} \quad S(q) > r \\ \text{into} & \\ \sum_i a_i(r)\cdot p_i + \sum_j b_j(r) + \tau(r)\cdot q & \end{array}$$

In this way we obtain the set of *ordered* terms which is denoted by $\mathcal{B}^{ord}$.

If p is an *ordered* term starting after s we may write $p \in \mathcal{B}^{ord}(s)$. If we take $\sum_{i\in\emptyset} p_i \equiv \delta$, then every *ordered* term is of the form

$$\sum_i a_i(r)\cdot p_i + \sum_j b_j(r) \quad \text{with } p_i \in \mathcal{B}^{ord}(r)$$

For each $s \in \mathbb{R}^{\geq 0}$ and *timed weak* bisimulation relation R we define a function f_R^s which maps a pair of *ordered* terms onto another pair of *ordered* terms. The construction of $f_R^s(p,q)$ guarantees that

Lemma 4.2.1 *If p,q in $\mathcal{B}^{ord}(s)$ and $(< s \gg p, T >, < s \gg q, T >)$ in R and $f_R^s(p,q) = (p',q')$, then*

- *BPA$\rho\delta\tau$ $\vdash a(s)\cdot p = a(s)\cdot p'$, $\quad a(s)\cdot q = a(s)\cdot q'$*
- *$p' \leftrightarrows q'$*

Here, $\leftrightarrows$ denotes strong bisimulation in the transition system without *implicit* τ-rule, which is completely characterized by BPA$\rho\delta$. The complexity of a pair of *ordered* terms is the pair of natural numbers ($depth(p) + depth(q)$, number of summands in $(p+q)$), and we assume a lexicographic ordering on pairs of natural numbers.

Definition 4.2.2

$$f^s_R \in \mathcal{B}^{ord}(s) \times \mathcal{B}^{ord}(s) \longrightarrow \mathcal{B}^{ord}(s) \times \mathcal{B}^{ord}(s)$$

We construct $f^s_R(p,q)$ inductively; we assume that we have constructed already the function f^t_R on pairs with smaller complexity for arbitrary t.

Consider (p,q) both in $\mathcal{B}^{ord}(s)$ and a *timed weak* bisimulation relation R containing $(< s \gg p, T >, < s \gg q, T >)$ such that

$$\begin{array}{rcl} p & \simeq & \sum_{j\in J} a_j(t)\cdot p_j + \sum_{j\in J'} a'_j(t) \\ q & \simeq & \sum_{l\in L} b_l(r)\cdot q_l + \sum_{l\in L'} b'_l(r) \end{array}$$

There are two cases, either $J = \emptyset = L$ or $J \neq \emptyset \vee L \neq \emptyset$. In the first case we simply take $f^s_R(p,q) = (p,q)$. The second case has on its turn two subcases, $t = r$ and $t \neq r$; these subcases are considered below. We assume that $J \neq \emptyset$.

- If $t = r$ then for every j in J $a_j \in A_\tau$ and there is a z_j (with $z_j \equiv q_l$ for some $l \in L$) such that

$$\begin{array}{ccc} < s \gg p, T > & \xrightarrow{a_j(t)} & < t \gg p_j, T > \\ R & & R \\ < s \gg q, T > & \xrightarrow{a_j(t)} & < t \gg z_j, T > \end{array}$$

  By induction we have already constructed $f^t_R(p_j, z_j) = (p'_j, q''_j)$. Similarly for every l we can find a term z'_l obtaining $f^t_R(q_l, z'_l) = (q'_l, p''_l)$. (It is more efficient of course to construct (q'_l, p''_l) only for those $l \in L$ such that $q_l \not\equiv z_j$ for every $j \in J$, but this will not be considered any further).

  Since $(< s \gg p, T >, < s \gg q, T >) \in R$ it is guaranteed that every terminating $a'_j(t)$ step can be matched by some terminating $b'_l(r)$ and vice versa.

  We define

$$f^s_R(p,q) \;=\; \left(\begin{array}{l} \sum_{j\in J} a_j(t)\cdot p'_j + \sum_{l\in L} b_l(t)\cdot p''_k + \sum_{j\in J'} a'_j(t) \\ \sum_{j\in J} a_j(t)\cdot q''_j + \sum_{l\in L} b_l(t)\cdot q'_l + \sum_{l\in L'} b'_l(t) \end{array} \right. , \left. \begin{array}{l} \\ \\ \end{array} \right)$$

- If $t \neq r$ then we may assume $t < r$. Then it must be the case that $a_j = \tau$ for all j in J and $J' = \emptyset$. We have for every j that $(< t \gg p_j, T >, < s \gg q, T >)$ in R. Since $s < t$ and $q \in \mathcal{B}^{ord}(t)$ we may extend R such that it remains a *timed weak* bisimulation relation which contains $(< s \gg q, T >, < t \gg q, T >)$. Hence we may extend R further such that it contains $(< t \gg p_j, T >, < t \gg q, T >)$ for each $j \in J$. By induction we have already constructed $f^t_R(p_j, q) = (p'_j, q''_j)$. We define

$$f^s_R(p,q) \;=\; \left(\sum_{j\in J} \tau(t)\cdot p'_j,\; \sum_{j\in J} \tau(t)\cdot p''_j \right. \left. \begin{array}{l} \\ \end{array} \right)$$

Now we are ready to give the completeness proof.

Theorem 4.2.3 *Completeness of $BPA\rho\delta\tau$*

$$p \leftrightarrows q \Longrightarrow BPA\rho\delta\tau \;\vdash p = q$$

Proof. We prove it first for *ordered* terms p and q, assume

$$\begin{array}{rcl} p & \simeq & \sum_{j\in J} a_j(t_j)\cdot p_j + \sum_{j\in J'} a'_j(t'_j) \\ q & \simeq & \sum_{l\in L} b_l(r_l)\cdot q_l + \sum_{l\in L'} b'_l(r'_l) \end{array}$$

We take some *timed weak* bisimulation relation R which contains $(< p, F >, < q, F >)$. From a root state (a state with boolean value F) no *implicit* τ steps have to be considered. Hence, for each j in J there is a l_j in L such that

$$a_j = b_{l_j} \quad \wedge \quad t_j = r_{l_j} \quad \wedge \quad (< t_j \gg p_j, T >, < t_j \gg q_{l_j}, T >) \in R$$

and we take $f_R^{t_j}(p_j, q_{l_j}) = (p'_j, q''_j)$. We do similar for each l in L and its associated index j_l in J. We construct p' and q',

$$\begin{array}{rcl} p' & \simeq & \sum_{j\in J} a_j(t_j)\cdot p'_j + \sum_{l\in L} b_l(r_l)\cdot p''_l + \sum_{j\in J'} a'_j(t'_j) \\ q' & \simeq & \sum_{j\in J} a_j(t_j)\cdot q''_j + \sum_{l\in L} b_l(r_l)\cdot q'_l + \sum_{l\in L'} b'_l(r'_l) \end{array}$$

such that $\mathrm{BPA}\rho\delta\tau \vdash p = p', q = q'$. By construction $p' \leftrightarrows q'$. Hence, $\mathrm{BPA}\rho\delta \vdash p' = q'$ and $\mathrm{BPA}\rho\delta\tau \vdash p = q$ follows immediately.

If we start with *non ordered* terms p and q, with $p \leftrightarrows q$, then we construct *ordered* terms p_o and q_o such that $\mathrm{BPA}\rho\delta \vdash p = p_0, q = q_o$. By soundness of $\mathrm{BPA}\rho\delta$ we obtain $p \leftrightarrows p_o$ and $q \leftrightarrows q_o$ and by transitivity of $\leftrightarrows$ we get $p_o \leftrightarrows q_o$. Now we have reduced it to the previous case and we conclude $\mathrm{BPA}\rho\delta\tau \vdash p_o = q_o$ from which we conclude $\mathrm{BPA}\rho\delta\tau \vdash p = q$. □

5 A Third Corresponding Semantics

In the two transition relations of the previous sections we had to keep track of a boolean value in each state to distinguish root states from internal states. In this Section we give another solution, we extend the set of terms $\mathcal{T}$ to $\mathcal{T}^{\gg}$ by adding a new *(absolute) time shift* operator $\gg$, which will be used to encode whether a state is internal or not. We saturate the transition relation with all possible *implicit* τ steps as was the case in the original semantics of Baeten and Bergstra as well. This enables us to deal with strong bisimulation instead of *timed weak* bisimulation.

Furthermore, we define only one relation $\longrightarrow \subset \mathcal{T}^{\gg} \times \mathcal{T}^{\gg}$. We now have $a(r) \xrightarrow{a(r)} \delta$ instead of $a(r) \xrightarrow{a(r)} \surd$ (which abbreviated $(a(r), r) \in Terminate$). By doing so we avoid a lot of rules. The action rules are given in Table 5. Two terms are bisimilar, denoted by $p \leftrightarrows_{\gg} q$, if there is a strong bisimulation relation $R \subset \mathcal{T}^{\gg} \times \mathcal{T}^{\gg}$. This approach only works when there are no occurrences of $\gg$ at the root level. Therefore we have the following equivalence for terms without $\gg$.

Lemma 5.0.4 $p, q \in \mathcal{T}$ $\qquad p \leftrightarrows_{\gg} q \iff p \leftrightarrows q$

We still have to prove the following Theorem, which can be proven easily for the bisimulation equivalence defined by the transition relation of Table 5.

$$a(r) \xrightarrow{a(r)} \delta \qquad \frac{L(p) < U(p)}{p \xrightarrow{\delta(U(p))} \delta} \qquad \frac{s < r \quad p \xrightarrow{a(r)} p'}{s \gg p \xrightarrow{a(r)} p'}$$

$$\frac{p \xrightarrow{a(r)} \delta}{p \cdot q \xrightarrow{a(r)} r \overline{\gg} q} \qquad \frac{p' \not\equiv \delta \quad p \xrightarrow{a(r)} p'}{p \cdot q \xrightarrow{a(r)} p' \cdot q} \qquad \frac{p \xrightarrow{a(r)} p'}{p + q \xrightarrow{a(r)} p' \,,\; q + p \xrightarrow{a(r)} p'}$$

$$\frac{s \gg p \xrightarrow{b(r)} p'}{s \overline{\gg} p \xrightarrow{b(r)} p'} \qquad \frac{r < s < U(r \gg p)}{r \overline{\gg} p \xrightarrow{\tau(s)} s \overline{\gg} p} \qquad \frac{s \overline{\gg} p \xrightarrow{\tau(t)} p' \quad p' \xrightarrow{a(r)} p''}{s \overline{\gg} p \xrightarrow{a(r)} p''}$$

Table 5: An Operational Semantics without Boolean Value ($a \in A_\tau$, $b \in A_{\delta\tau}$, $r, s > 0$)

Lemma 5.0.5 *Bisimulation Equivalence is a congruence w.r.t. all operators*

Proof. If all action rules of a Transition System Specification are in Groote's *ntyft/ntyxt* format, then bisimulation equivalence is a congruence (see [Gro89]). We have to prove only for the closures rules that they can be written into this format, since in [Klu91] we proved it already for all other action rules of Table 5. The first and the third closure rule are already in the right format and for the second one we may take

$$\frac{s < t \quad r \gg p \xrightarrow{b(t)} p''}{r \overline{\gg} p \xrightarrow{\tau(s)} s \overline{\gg} p} \quad \text{instead of} \quad \frac{r < s < U(r \gg p)}{r \overline{\gg} p \xrightarrow{\tau(s)} s \overline{\gg} p}$$

□

In this paper we will not mention anymore this variant of the semantics, since we do not want to discuss the inclusion of the operator $\overline{\gg}$ in the theory.

6 Symbolic Processes as Timed Processes

In this Section we interpret each symbolic process term as a timed process. We will investigate the resulting subtheory.

6.1 The Interpretation of Symbolic Process Terms as Timed Processes

By using the integral construct of Baeten and Bergstra we can express a process which executes an a somewhere in time by the process term $\int_{v>0} a(v)$. The formal introduction of the integral construct is postponed till the next section. By extending the syntax with the

integral construct we obtain BPA$\rho\delta\tau I$. We define a function RT : BPA$\delta\tau \longrightarrow$ BPA$\rho\delta\tau I$, which interprets every symbolic process term as a timed process.

$$\begin{array}{lll} a \in A & RT(a) & \stackrel{def}{=} \int_{v>0} a(v) \cdot \int_{w>0} \tau(w) \\ & RT(\tau) & \stackrel{def}{=} \int_{v>0} \tau(v) \\ & RT(\delta) & \stackrel{def}{=} \int_{v>0} \delta(v) \\ & RT(p+q) & \stackrel{def}{=} RT(p) + RT(q) \\ & RT(p \cdot q) & \stackrel{def}{=} RT(p) \cdot RT(q) \end{array}$$

Originally, Baeten and Bergstra had $RT(a) \stackrel{def}{=} \int_{v>0} a(v)$ (they denote $RT(p)$ by $\underline{p}$). But in this case the first τ-law $X \cdot \tau = X$ ([Mil89]) would not be sound anymore. Since we prefer to maintain this law we define $RT(a) \stackrel{def}{=} \int_{v>0} a(v) \cdot \int_{w>0} \tau(w)$.

The range of the function RT is denoted by $RT(\text{BPA}\delta\tau)$.

6.2 A Delay Bisimulation Semantics on BPA$\delta\tau$

Here we give a semantics for symbolic processes which corresponds with the semantics of their timed interpretations. In the previous sections we have studied transition relations which where τ-saturated, i.e all possible appropriate τ-steps between internal states were added.

In this Section we will not saturate the transition relation but we move the τ-saturation into the definition of bisimulation. In this way we obtain a notion of delay bisimulation [GW89].

$$a \xrightarrow{a} \surd \qquad \frac{p \xrightarrow{a} \surd}{p+q \xrightarrow{a} \surd} \qquad \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'}$$

$$\frac{p \xrightarrow{a} \surd}{p \cdot q \xrightarrow{a} q} \qquad \frac{p \xrightarrow{a} p'}{p \cdot q \xrightarrow{a} p' \cdot q}$$

Table 6: Transition System Specification for Symbolic Process Terms ($a \in A_\tau$)

Moreover we want to get rid of this boolean value. In the previous sections the boolean value guaranteed that the closure rules were only applied on internal states, hence from a root only explicit steps were defined. This property will be expressed now by a root condition which is stronger then the usual one ([GW89]).

If there is a path $p_0 \xrightarrow{\tau} p_1 \ldots \xrightarrow{\tau} p_k$ then we may write $p_0 \Longrightarrow p_k$. In this Section we allow ourselves the freedom to consider $\surd$ as a special state.

Definition 6.2.1 *A relation* $R \subset \text{BPA}\delta\tau \cup \{\surd\} \times \text{BPA}\delta\tau \cup \{\surd\}$ *is strongly rooted w.r.t. p and q if it obeys the following* ($a \in A_\tau$)*:*

- *R relates p and q with each other and not with other terms.*
- $p \xrightarrow{a} p'$ *implies that there is a* q' *such that* $q \xrightarrow{a} q'$ *and* $(p', q') \in R$.
- $q \xrightarrow{a} q'$ *implies that there is a* p' *such that* $p \xrightarrow{a} p'$ *and* $(q', p') \in R$.

A relation R is *rooted* if it satisfies the first of the conditions above. An example of a relation which is *strongly rooted* is given in Figure 4. In this Figure we can see that the *strongly rooted* requirement corresponds to the requirement that the closure rule is applied on internal states only.

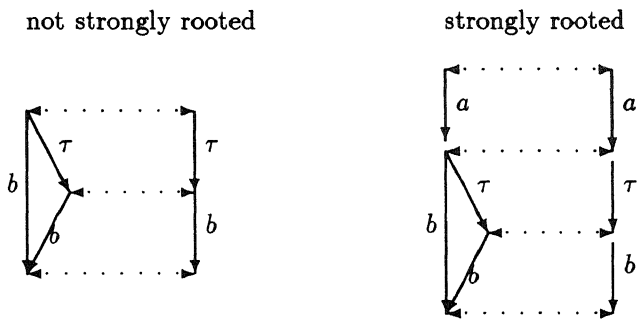


Figure 4: Example of Strongly Rootedness

Definition 6.2.2 *Symbolic Bisimulation or Strongly Rooted Delay Bisimulation*

$$p \leftrightarrows_{sym} q$$

iff there is a symmetric relation $R \subset \mathrm{BPA}\delta\tau \cup \{\surd\} \times \mathrm{BPA}\delta\tau \cup \{\surd\}$ *which is strongly rooted w.r.t. p and q such that for every pair* (r, s) *in R with* $r \xrightarrow{a} r'$ *either* $a = \tau$ *and* (r', s) *in R or* $s \Longrightarrow s'' \xrightarrow{a} s'$ *and* (r', s') *in R .*

In the sequel we will use the definition of rooted Branching Bisimulation ([GW89]) as well.

Definition 6.2.3 *Rooted Branching Bisimulation*

$$p \leftrightarrows_{rb} q$$

iff there is a bisimulation relation $R \subset \mathrm{BPA}\delta\tau \times \mathrm{BPA}\delta\tau$ *which is rooted w.r.t. p and q such that for every pair* (r, s) *in R with* $r \xrightarrow{a} r'$ *either* $a = \tau$ *and* (r', s) *in R or* $s \Longrightarrow s'' \xrightarrow{a} s'$ *and both* (r, s'') *and* (r', s') *in R.*

We have the following Lemma, since strongly rootedness is implied by rooted branching bisimulation.

Lemma 6.2.4 $\quad p \leftrightarrows_{rb} q \quad \Longrightarrow \quad p \leftrightarrows_{sym} q$

Finally we have the following Theorem which states that the equivalence $\leftrightarrows_{sym}$ is exactly the equivalence which we obtain by interpreting symbolic processes as timed processes. Bisimulation equivalence over BPA$\delta\tau$ is denoted by $\leftrightarrows_{sym}$ and bisimulation equivalence over RT(BPA$\delta\tau$) is denoted by $\leftrightarrows_{time}$.

Theorem 6.2.5 $\quad p \leftrightarrows_{sym} q \iff RT(p) \leftrightarrows_{time} RT(q)$

6.3 Completeness for Strongly Rooted Delay Bisimulation

In this Section we give a complete axiomatization for the equivalence $\leftrightarrows_{sym}$ following the work of Van Glabbeek and Weijland ([GW89]). We take the axioms DEL 1 and 2 from [GW89], where it is proven that these two axioms characterize Branching Bisimulation completely. Furthermore, we take the axiom

$$\tau \cdot Y = \tau \cdot Y + Y \,,$$

which can be found in [GW89] as well and we require that it is only applied in a context. In this way we obtain

$$\text{DEL3} \quad X \cdot (\tau \cdot Y + Z) = X \cdot (\tau \cdot Y + Y + Z)$$

Since Van Glabbeek and Weijland use a graph model we have to define the following. The set of graphs $\mathcal{G}$ is the set of triples (N, E, r) where N is a set of nodes , $E \subset N \times N$ is a set of edges and $r \in N$ is the root. Moreover, if $g = (N, E, r)$ then $E(g) = E$. A graph g is trivial if $E(g) = \emptyset$. $\mathcal{G}^+$ is the set of non trivial graphs. The graph of a term p is denoted by $[\![p]\!]$, this graph can be seen as that part of the transition relation which is associated with p, where $\surd$ is now considered as a special state.

Definition 6.3.1 *The graph rewriting* $\longrightarrow_\tau$
If a graph g has a path $s \xrightarrow{\tau} s' \xrightarrow{a} s''$ where s is not the root of g and g has no edge $s \xrightarrow{a} s''$ then $s \xrightarrow{a} s''$ is added.

Lemma 6.3.2 $\quad [\![p]\!] \longrightarrow_\tau g \implies \exists p' \quad [\![p']\!] = g \wedge \text{DEL3} \vdash p = p'$

As in [GW89] we can prove easily

Lemma 6.3.3

DEL1 $\quad X \cdot \tau = X$
DEL2 $\quad X \cdot (\tau \cdot (Y + Z) + Y) = X \cdot (Y + Z)$
DEL3 $\quad X \cdot (\tau \cdot Y + Z) = X \cdot (\tau \cdot Y + Y + Z)$

Table 7: BPA$\delta\tau_{del}$

- *Both $\mathcal{G}$ and $\mathcal{G}^+$ are closed under $\longrightarrow_\tau$.*
- *$\longrightarrow_\tau$ is confluent and terminating.*

Definition 6.3.4 *A graph g is τ-saturated if it can not be reduced any further by $\longrightarrow_\tau$.*

Lemma 6.3.5 *If g and h are τ-saturated graphs then*

$$g \leftrightarrows_{sym} h \iff g \leftrightarrows_{rb} h$$

Proof. Since $\leftrightarrows_{rb}$ is a smaller equivalence than $\leftrightarrows_{sym}$ it is sufficient to prove that $R : g \leftrightarrows_{sym} h$ implies $R : g \leftrightarrows_{rb} h$.

- The roots are related only with each other
- if $R(r,s)$ and $r \xrightarrow{a} r'$ then either
  - $a = \tau$ and $(r',s) \in R$ or
  - $s \Rightarrow s'' \xrightarrow{a} s'$ with $(r',s') \in R$, assume $s = s_0 \xrightarrow{\tau} s_1 \ldots \xrightarrow{\tau} s_k = s''$, since h is τ-*saturated* we have $s_i \xrightarrow{a} s'$ and also $s \xrightarrow{a} s'$. □

Theorem 6.3.6 $$p \leftrightarrows_{sym} q \iff \mathrm{BPA}\delta\tau_{del} \vdash p = q$$

Proof.

- $\Longleftarrow$ The soundness can be seen by investigating the operational semantics.
- $\Longrightarrow$ By construction of the graph model we have $p \leftrightarrows_{sym} q$ iff $[\![p]\!] \leftrightarrows_{sym} [\![q]\!]$. By τ-*saturating* the graphs $[\![p]\!]$ and $[\![q]\!]$ we obtain g and h. By Lemma 6.3.2 we can construct terms p' and q' such that $[\![p']\!] = g$, $[\![q']\!] = h$ and $\mathrm{BPA}\delta\tau_{del} \vdash p = p', q = q'$. By transitivity of $\leftrightarrows_{sym}$ we get $g \leftrightarrows_{sym} h$ and since g and h are τ-*saturated* we obtain $g \leftrightarrows_{rb} g$ and by the completeness result of Van Glabbeek and Weijland we conclude $\mathrm{BPA}\delta\tau_{del} \vdash p' = q'$, from which it follows directly that $\mathrm{BPA}\delta\tau_{del} \vdash p = q$. □

7 Adding Integrals

An *integral* can be considered as a sum over a continuum of alternatives, this notion is introduced in [BB91]. Baeten and Bergstra allow integration over arbitrary subsets of the real numbers and they allow more then one integral behind each other. The idea of *prefixed integration* is that every action has as time stamp a *time variable* v taken from some set $TVar$, and the action is directly preceded by the integral binding this v. Moreover, only *Intervals* are allowed. In [Klu91] a completeness result is given for *prefixed integration*. The term $\int_{v \in <0,1>} a(v)$ denotes the process which executes an action a somewhere between 0 and 1. An *integral* binds a *time variable*, which may occur in the rest of the term, for example the term $\int_{v \in <0,1>} a(v) \cdot \int_{w \in <v+1,v+2>} b(w)$ denotes the process which executes an action a at t where t is within 0 and 1. It waits between 1 and 2 time

units after t and executes an action b. Hence, the *bounds* of a interval of an *integral* are (linear) expressions over the real numbers. Let $t \in \mathbb{R}^{\geq 0}, v \in TVar$ then we can define the set *Bounds* as follows:

$$b \in Bounds \quad b := t \mid v \mid b_1 + b_2 \mid b_1 \dot{-} b_2 \mid t \cdot b$$

where $\dot{-}$ denotes the monus operator, i.e $5 \dot{-} 3 = 2$ but $3 \dot{-} 5 = 0$. If $b \in Bounds$ then the set of *time variables* occurring in b is denoted by $tvar(b)$. Now we can construct intervals like $< 1, 9 >$ and $< v + 3, w >$. An interval without free *time variables* can be considered as a connected part of the nonnegative reals. However, we don't want to deal with the complexity of set theory over reals and we want to define intervals containing occurrences of free *time variables*. Hence, every interval is a four tuple, containing two booleans and two reals. The interval $V = (F, 1, 2, T)$ is abbreviated in the sequel by $V =< 1, 2]$, denoting that the lower bound is open and 1, and that the upper bound is closed and 2. If $b \in Bounds$ then $b \in V$ denotes the logical expression of (in)equalities

TAUI1 $V < W < U(X) \wedge U(Y) \leq inf(W)$
$\int_{v \in V} a(v) \cdot ((\int_{w \in W} \tau(w)) \cdot X + Y) = \int_{v \in V} a(v) \cdot (inf(W) \gg X + Y)$
TAUI2 $V < W < U(X + Y) \wedge U(Y) = sup(W)$
$\int_{v \in V} a(v) \cdot ((\int_{w \in W} \tau(w)) \cdot (X + Y) + X) = \int_{v \in V} a(v) \cdot (X + inf(W) \gg Y)$
TAUI3 $V < W < U(X) \wedge U(X) = sup(W)$
$\int_{v \in V} a(v) \cdot ((\int_{w \in W} \tau(w)) \cdot X + Y) = X \cdot ((\int_{w \in W} \tau(w)) \cdot X + inf(W) \gg X + Y)$

Table 8: $\mathrm{BPA}\rho I = \mathrm{BPA}\rho\delta\tau + TAUI1 - 3$

$1 < b \leq 2$. Similarly we have $t \in V_1 \cup V_2$, $t < sup(V)$, $V = \emptyset$, $V_1 < V_2$ and $V < t$ as abbreviations for logical expressions over *Bounds*.

We can redefine the set of terms. Let $a \in A_{\delta\tau}$, $V \in Int$, $v \in TVar$, $b \in Bounds$

$$p \in \mathcal{T} \quad p := \int_{v \in V}(a(v)) \mid \int_{v \in V}(a(v) \cdot p) \mid p \cdot q \mid p + q \mid r \gg p$$

We abbreviate $\int_{v \in [w,w]} a(v)$ by $a(w)$ and $\delta(0)$ by δ. In this definition the notion of prefixed integration becomes clear; every action has as time stamp a *time variable* v and is directly preceded by its binding integral. Hence, we do not allow a term like $\int_{v \in V} \int_{w \in W}(a(v) \cdot \int_{r \in <v,w>} b(1) \cdot c(r))$. On these terms we have notions as $FV()$ for the set of free *time variables* , α-conversion, and substitution. If a term or interval has no free *time variables*, then it is called *time closed*.

The adaptions of the transition relation as given in Table 3 are given in Table 9. Due to the integral construct there is no more a discrete notion of a moment of choice, hence the closure rule must generate infinitely many τ-steps.

Theorem 7.0.7 *The laws* TAUI1 – 3 *are sound*

$$r \in V \quad < \int_{v \in V} a(v), b > \xrightarrow{a(r)} \surd \quad \text{instead of} \quad < a(r), b > \xrightarrow{a(r)} \surd$$

$$r \in V \quad < \int_{v \in V} (a(v) \cdot p), b > \xrightarrow{a(r)} r \gg p[r/v]$$

$$\frac{r < s < U(s \gg p)}{< r \gg p, T > \xrightarrow{\tau(s)} < s \gg p, T >} \quad \text{instead of} \quad \frac{s = S(r \gg p) < U(s \gg p)}{< r \gg p, T > \xrightarrow{\tau(s)} < s \gg p, T >}$$

Table 9: (Additional/Changed) Action Rules for Integral Construct

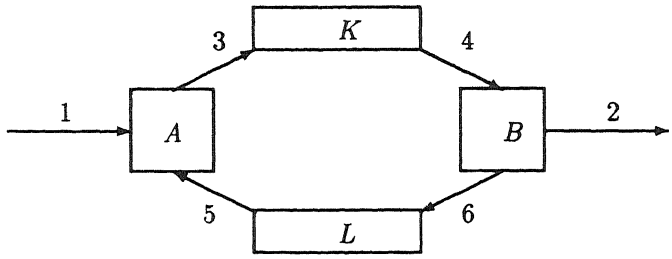
8 Protocol Verification

Now we are ready to verify a protocol which is time dependent. First we have to state the fact that every time guarded recursive specification has exactly one solution (RSP and RDP). A guarded recursive specification is time guarded if there is a lower bound bound on the time interval between two recursion variable unfoldings. This is only an informal characterization but it is needed to exclude so-called 'Zeno' machines. The proof of this principle and a thorough treatment of time guarded specifications do not fall into the scope of this paper, they will treated in later papers. Next, we have to state an *Unwind Principle* (UP) which allows us to unwind a recursive specification infinitely many times. The dots in the derivation below express that this principle is not provable within the theory BPA$\rho\delta\tau$ in a finite derivation . We take $Y(t)$ such that

$$t \gg Y(t) = Y(t)$$

thus assuming that $Y(t)$ has no parts starting at or before t. We define $X(t)$ as

$$X(t) = \tau(t) \cdot \{Y(t+s) + \tau(t+r_0) \cdot X(t+r_1)\}$$



for some $r_0 < r_1$ and $r_0 < s$ then

$$\begin{aligned}
X(t) &= \tau(t)\cdot\{Y(t+s)+\tau(t+r_0)\cdot X(t+r_1)\}\\
&= \tau(t)\cdot\{\tau(t+r_0)\cdot Y(t+s)+X(t+r_1)\}\\
&= \tau(t)\cdot\{\tau(t+r_0)\cdot Y(t+s)+\\
&\qquad \tau(t+r_1)\cdot\{Y(t+s+r_1)+\tau(t+r_0+r_1)\cdot X(t+2\cdot r_1)\}\}\\
&= \tau(t)\cdot\{\tau(t+r_0)\cdot Y(t+s)+\{Y(t+s+r_1)+\tau(t+r_0+r_1)\cdot X(t+2\cdot r_1)\}\}\\
&= \tau(t)\cdot\{\tau(t+r_0)\cdot Y(t+s)+\tau(t+r_0+r_1)\cdot Y(t+s+r_1)+X(t+2\cdot r_1)\}\\
&\ \ \vdots\\
&\overset{UP}{=} \tau(t)\cdot\{\textstyle\sum_{n=0}^{\infty}\tau(t+r_0+n\cdot r_1)\cdot Y(t+s+n\cdot r_1)\}
\end{aligned}$$

We take example 6.15 of [BB91], which is a PAR protocol (Positive Acknowledgement with Retransmission). Some small changes are made. In the example below the set H contains all read and send actions along the internal ports 3,4,5 and 6. The operator δ_H renames every action which occurs in H to δ. It is known as the *encapsulation* operator and it forces actions to communicate, for example if $r_i|s_i = c_i$ then $\delta_{\{r_i,s_i\}}(s_i\|r_i) = c_i$.

8.1 The Specification and the Implementation of the Protocol

First we define the individual components.

$$\begin{aligned}
A &= A(0,0)\\
A(b,t) &= \textstyle\sum_{d\in D}\int_{v>t} r_1(d)(v)\cdot A(b,d,t)\\
A(b,d,t) &= s_3(db)(t+0.001)\cdot\{\textstyle\int_{w\in[t+0.001,t+0.01>} r_5(ack)(w)\cdot A(1-b,w)+\\
&\qquad time_out(t+0.01)\cdot A(b,d,t+0.01)\}\\
\\
K &= \textstyle\sum_{f\in D\times B}\int_{v>0} r_3(f)(v)\cdot\{s_4(f)(v+0.002)+error_K(v+0.001)\}\cdot K\\
L &= \textstyle\int_{v>0} r_6(ack)(v)\cdot\{s_5(ack)(v+0.002)+error_L(v+0.001)\}\cdot L\\
\\
B &= B(0)\\
B(b) &= \textstyle\sum_{d\in D}\int_{v>0} r_4(db)(v)\cdot s_2(d)(v+0.001)\cdot B(1-b,v)+\\
&\qquad \textstyle\sum_{d\in D}\int_{v>0} r_4(d(b-1))(v)\cdot B(b,v)\\
B(b,t) &= s_6(ack)(t+0.002)\cdot B(b)
\end{aligned}$$

The implementation of the protocol is the following merge:

$$PAR_{impl} = \delta_H(A \parallel K \parallel L \parallel B)$$

8.2 Expanding the Definitions

We expand the definitions, for each new configuration a new recursion variable is chosen. In this way we obtain the parameterized recursion variables $X_0 - X_4$ and $Y_1 - Y_4$.

$$PAR_{impl} = X_0(0,0)$$

$$\begin{aligned} X_0(b,t_0) &= \delta_H(A(b,t_0)\|K\|L\|B(b)) \\ &= \int_{v>t_0} \sum_{d\in D} r_1(d)(v) \cdot X_1(b,d,v) \end{aligned}$$

$$\begin{aligned} X_1(b,d,t) &= \delta_H\ (\ A(b,d,t)\|\ K\ \|\ L\ \|\ B(b)\ ) \\ &= \delta_H\ (\ \underline{s_3(db)(t+0.001)} \cdot \\ &\qquad \underline{[\int_{w\in[t,t+0.01)} r_5(ack,w) \cdot A(1-b,w)\ +\ time\_out(t+0.01) \cdot A(b,d,t+0.01)]} \\ &\qquad \|\ \textstyle\sum_{f\in D\times B} \int_{v>0} \underline{r_3(f)(v)} \cdot [s_4(f)(v+0.002) + error_K(v+0.001)] \cdot K \\ &\qquad \|\ L \\ &\qquad \|\ B(b) \\ &\qquad) \\ &= c_3(\ db)(t+0.001)\ \cdot\ X_2(b,d,t) \end{aligned}$$

$$\begin{aligned} X_2(b,d,t) &= \delta_H\ (\ \underline{[\int_{w\in[t,t+0.01)} r_5(ack,w) \cdot A(1-b,w)\ +\ time\_out(t+0.01)} \cdot A(b,d,t+0.01)] \\ &\qquad \|\ \underline{[s_4(db)(t+0.003) + error_K(t+0.002)]} \cdot K \\ &\qquad \|\ L \\ &\qquad \|\ \textstyle\sum_{d\in D} \int_{v>0} \underline{r_4(db)(v)} \cdot \underline{s_2(d)(v+0.001)} \cdot B(1-b,v) + \\ &\qquad\quad \textstyle\sum_{d\in D} \int_{v>0} r_4(d(1-b))(v) \cdot B(b,v) \\ &\qquad) \\ &= c_4(\ db)(t+0.003) \cdot s_2(d)(t+0.004) \cdot X_3(b,d,t) + \\ &\qquad error_K(t+0.002) \cdot time\_out(t+0.01) \cdot X_1(b,d,t+0.01) \end{aligned}$$

$$\begin{aligned} X_3(b,d,t) &= \delta_H\ (\ [\int_{w\in[t,t+0.01)} r_5(ack,w) \cdot A(1-b,w)\ +\ time\_out(t+0.01) \cdot A(b,d,t+0.01)] \\ &\qquad \|\ K \\ &\qquad \|\ \int_{v>0} \underline{r_6(ack)(v)}\ \cdot\ [s_5(ack)(v+0.002) + error_L(v+0.001)]\ \cdot L \\ &\qquad \|\ \underline{s_6(ack)(v)(t+0.005)} \cdot B(1-b) \\ &\qquad) \\ &= c_6(\ ack)(t+0.005) \cdot X_4(b,d,t) \end{aligned}$$

$$\begin{aligned} X_4(b,d,t) &= \delta_H\ (\ [\int_{w\in[t,t+0.01)} \underline{r_5(ack,w)} \cdot A(1-b,w)\ +\ \underline{time\_out(t+0.01)} \cdot A(b,d,t+0.01)] \\ &\qquad \|\ K \\ &\qquad \|\ \underline{[s_5(ack)(t+0.007) + error_L(t+0.006)]}\ \cdot L \\ &\qquad \|\ B(1-b) \\ &\qquad) \\ &= c_5(\ ack)(t+0.007) \cdot X_0(1-b,t+0.007) + \\ &\qquad error_L(t+0.006) \cdot time\_out(t+0.01) \cdot Y_1(b,d,t+0.01) \end{aligned}$$

$$\begin{aligned} Y_1(b,d,t) &= \delta_H\ (\ A(b,d,t)\|\ K\ \|\ L\ \|\ B(1-b)\ ) \\ &= \delta_H\ (\ \underline{s_3(db)(t+0.001)} \cdot \\ &\qquad \underline{[\int_{w\in[t,t+0.01)} r_5(ack,w) \cdot A(1-b,w)\ +\ time\_out(t+0.01) \cdot A(b,d,t+0.01)]} \end{aligned}$$

$$
\begin{aligned}
&\| \textstyle\sum_{f\in D\times B} \int_{v>0} \underline{r_3(f)(v)} \cdot [s_4(f)(v+0.002) + error_K(v+0.001)] \cdot K \\
&\| L \\
&\| B(1-b) \\
&) \\
&= c_3(\,db)(t+0.001) \;\cdot\; Y_2(b,d,t)
\end{aligned}
$$

$$
\begin{aligned}
Y_2(b,d,t) \;&= \delta_H \,(\; [\textstyle\int_{w\in[t,t+0.01)} r_5(ack,w)\cdot A(1-b,w) \;+\; \underline{time\_out(t+0.01)} \cdot A(b,d,t+0.01)] \\
&\quad \| \underline{[s_4(f)(t+0.003)} + \underline{error_K(t+0.002)]} \cdot K \\
&\quad \| L \\
&\quad \| \textstyle\sum_{d\in D} \int_{v>0} \underline{r_4(d(1-b))(v)} \cdot \underline{s_2(d)(v+0.001)} \cdot B(b,v) + \\
&\quad\;\; \textstyle\sum_{d\in D} \int_{v>0} r_4(db))(v) \cdot B(1-b,v) \\
&\quad) \\
&= c_4(\,db)(t+0.003) \cdot Y_3(b,d,t) + \\
&\quad\; error_K(t+0.002) \cdot time\_out(t+0.01) \cdot Y_1(b,d,t+0.01)
\end{aligned}
$$

$$
\begin{aligned}
Y_3(b,d,t) \;&= \delta_H \,(\; [\textstyle\int_{w\in[t,t+0.01)} r_5(ack,w)\cdot A(1-b,w) \;+\; time\_out(t+0.01) \cdot A(b,d,t+0.01)] \\
&\quad \| K \\
&\quad \| \textstyle\int_{v>0} \underline{r_6(ack)(v)} \;\cdot\; [s_5(ack)(v+0.002) + error_L(v+0.001)] \;\cdot L \\
&\quad \| \underline{s_6(ack)(v)(t+0.005)} \cdot B(b) \\
&\quad) \\
&= c_6(\,ack)(t+0.005) \cdot Y_4(b,d,t)
\end{aligned}
$$

$$
\begin{aligned}
Y_4(b,d,t) \;&= \delta_H \,(\; [\textstyle\int_{w\in[t,t+0.01)} \underline{r_5(ack,w)}\cdot A(1-b,w) \;+\; \underline{time\_out(t+0.01)} \cdot A(b,d,t+0.01)] \\
&\quad \| K \\
&\quad \| \underline{[s_5(ack)(t+0.007)} + \underline{error_L(t+0.006)]} \;\cdot L \\
&\quad \| B(b) \\
&\quad) \\
&= c_5(\,ack)(t+0.007) \cdot X_0(1-b,t+0.007) + \\
&\quad\; error_L(t+0.006) \cdot time\_out(t+0.01) \cdot Y_1(b,d,t+0.01)
\end{aligned}
$$

8.3 Abstracting from Internal Steps

We apply the renaming operator τ_I which renames every atomic action $a(t)$ to $\tau(t)$ except for the actions $r_1(d)(t)$ and $s_2(d)(t)$.

$$
\begin{aligned}
\tau_I(X_0(b,t_0)) &= \textstyle\int_{t>t_0} \sum_{d\in D} r_1(d)(t) \cdot \tau_I(X_1(b,d,t)) \\
\tau_I(X_1(b,d,t)) &= \tau(t+0.001) \;\cdot\; \tau_I(X_2(b,d,t)) \\
\tau_I(X_2(b,d,t)) &= \tau(t+0.003)\cdot s_2(d)(t+0.004)\cdot \tau_I(X_3(b,d,t)) + \\
&\quad\; \tau(t+0.002)\cdot\tau(t+0.01)\cdot\tau_I(X_1(b,d,t+0.01)) \\
\tau_I(X_3(b,d,t)) &= \tau(t+0.005)\cdot\tau_I(X_4(b,d,t)) \\
\tau_I(X_4(b,d,t)) &= \tau(t+0.007)\cdot\tau_I(X_0(1-b,t+0.007)) + \\
&\quad\; \tau(t+0.006)\cdot\tau(t+0.01)\cdot\tau_I(Y_1(b,d,t+0.01))
\end{aligned}
$$

$$
\begin{aligned}
\tau_I(Y_1(b,d,t)) &= \tau(t+0.001) \;\cdot\; \tau_I(Y_2(b,d,t)) \\
\tau_I(Y_2(b,d,t)) &= \tau(t+0.003)\cdot\tau_I(Y_3(b,d,t)) + \\
&\quad\; \tau(t+0.002)\cdot\tau(t+0.01)\cdot\tau_I(Y_1(b,d,t+0.01)) \\
\tau_I(Y_3(b,d,t)) &= \tau(t+0.005)\cdot\tau_I(Y_4(b,d,t))
\end{aligned}
$$

$$\begin{aligned}\tau_I(Y_4(b,d,t)) &= \tau(t+0.007)\cdot\tau_I(X_0(1-b,t+0.007)) + \\ &\quad \tau(t+0.006)\cdot\tau(t+0.01)\cdot\tau_I(Y_1(b,d,t+0.01))\end{aligned}$$

Now we can apply the τ-law and its implied identities (such as the τ-swap and the τ-removal).

$$\begin{aligned}\tau_I(X_1(b,d,t)) &= \tau(t+0.001)\ \cdot\ \tau_I(X_2(b,d,t)) \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \underline{\tau(t+0.003)}\cdot s_2(d)(t+0.004)\cdot\tau_I(X_3(b,d,t))\ + \\ &\quad\ \underline{\tau(t+0.002)}\cdot\underline{\tau(t+0.01)}\cdot\tau_I(X_1(b,d,t+0.01))\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ s_2(d)(t+0.004)\cdot\underline{\tau(t+0.005)}\cdot\tau_I(X_4(b,d,t))\ + \\ &\quad\ \tau(t+0.002)\cdot\tau_I(\overline{X_1(b,d,t+0.01)})\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ s_2(d)(t+0.004)\cdot \\ &\quad\ \{\ \underline{\tau(t+0.007)}\cdot\tau_I(X_0(1-b,t+0.007)) + \\ &\quad\ \ \underline{\tau(t+0.006)}\cdot\underline{\tau(t+0.01)}\cdot\tau_I(Y_1(b,d,t+0.01))\ \}\ + \\ &\quad\ \tau(t+0.002)\cdot\tau_I(\overline{X_1(b,d,t+0.01)})\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ s_2(d)(t+0.004)\cdot \\ &\quad\ \ \{\ \tau_I(X_0(1-b,t+0.007))\ +\tau(t+0.006)\cdot\tau_I(Y_1(b,d,t+0.01))\ \}\ + \\ &\quad\ \tau(t+0.002)\cdot\tau_I(X_1(b,d,t+0.01))\ \}\end{aligned}$$

$$\begin{aligned}\tau_I(Y_1(b,d,t)) &= \tau(t+0.001)\ \cdot\ \tau_I(Y_2(b,d,t)) \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \underline{\tau(t+0.003)}\cdot\tau_I(Y_3(b,d,t)) + \\ &\quad\ \underline{\tau(t+0.002)}\cdot\underline{\tau(t+0.01)}\cdot\tau_I(Y_1(b,d,t+0.01))\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \underline{\tau(t+0.005)}\cdot\tau_I(Y_4(b,d,t)) + \\ &\quad\ \underline{\tau(t+0.002)}\cdot\tau_I(Y_1(b,d,t+0.01))\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \{\ \underline{\tau(t+0.007)}\cdot\tau_I(X_0(1-b,t+0.007)) + \\ &\quad\ \ \underline{\tau(t+0.006)}\cdot\underline{\tau(t+0.01)}\cdot\tau_I(Y_1(b,d,t+0.01))\ \}\ + \\ &\quad\ \tau(t+0.002)\cdot\tau_I(\overline{Y_1(b,d,t+0.01)})\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \tau_I(X_0(1-b,t+0.007)) + \tau(t+0.006)\cdot\tau_I(Y_1(b,d,t+0.01)) + \\ &\quad\ \underline{\tau(t+0.002)\cdot\tau_I(Y_1(b,d,t+0.01))}\ \} \\ &= \tau(t+0.001)\cdot \\ &\quad \{\ \tau_I(X_0(1-b,t+0.007)) + \tau(t+0.006)\cdot\tau_I(Y_1(b,d,t+0.01))\ \}\end{aligned}$$

Summarizing,

$$\tau_I(X_0(b,t_0)) = \int_{t>t_0}\sum_{d\in D} r_1(d)(t)\cdot\tau_I(X_1(b,d,t))$$

$$\begin{aligned}\tau_I(X_1(b,d,t)) &= \tau(t+0.001)\cdot \\ &\quad \{\ s_2(d)(t+0.004)\cdot \\ &\quad\ \ \{\ \tau_I(X_0(1-b,t+0.007))\ +\tau(t+0.006)\cdot\tau_I(Y_1(b,d,t+0.01))\ \}\ +\end{aligned}$$

$$\tau(t+0.002)\cdot\tau_I(X_1(b,d,t+0.01))\ \}$$

$$\tau_I(Y_1(b,d,t)) \;=\; \tau(t+0.001)\cdot \tau_I(X_0(1-b,t+0.007))+\tau(t+0.006)\cdot\tau_I(Y_1(b,d,t+0.01))$$

By applying the *Unwind Principle*:

$$\begin{aligned}\tau_I(X_1(b,d,t)) &= \tau(t+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(t+0.002+n\cdot 0.01)\cdot s_2(d)(t+0.004+n\cdot 0.01)\cdot\\ &\quad\{\ \tau_I(X_0(1-b,t+0.007+n\cdot 0.01))+\\ &\quad\ \tau(t+0.006+n\cdot 0.01)\cdot\tau_I(Y_1(b,d,t+(n+1)\cdot 0.01))\ \}\end{aligned}$$

$$\begin{aligned}\tau_I(Y_1(b,d,t)) &= \tau(t+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(t+0.006+n\cdot 0.01)\\ &\quad\tau_I(X_0(1-b,t+0.007+n\cdot 0.01))\end{aligned}$$

Adding together

$$\begin{aligned}\tau_I(X_1(b,d,t)) &= \tau(t+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(t+0.002+n\cdot 0.01)\cdot s_2(d)(t+0.004+n\cdot 0.01)\cdot\\ &\quad\{\ \tau_I(X_0(1-b,t+0.007+n\cdot 0.01))+\\ &\quad\ \{\ \tau(t+0.006+n\cdot 0.01)\cdot\\ &\quad\ \{\textstyle\sum_{n'=0}^{\infty}\tau((t+(n+1)\cdot 0.01)+0.006+n'\cdot 0.01)\cdot\\ &\quad\ \tau_I(X_0(1-b,((t+(n+1)\cdot 0.01)+0.007+n'\cdot 0.01))\}\ \}\\ &= \tau(t+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(t+0.002+n\cdot 0.01)\cdot s_2(d)(t+0.004+n\cdot 0.01)\cdot\\ &\quad\{\ \tau(t+0.006+n\cdot 0.01)\ \cdot\ \tau_I(X_0(1-b,t+0.007+n\cdot 0.01))+\\ &\quad\ \{\textstyle\sum_{n'=0}^{\infty}\tau(t+0.006+(n+n'+1)\cdot 0.01)\cdot\\ &\quad\ \tau_I(X_0(1-b,(t+0.007+(n+n'+1)\cdot 0.01))\}\ \}\\ &= \tau(t+0.001)\cdot\textstyle\sum_{n=0}^{\infty}\tau(t+0.002+n\cdot 0.01)\cdot s_2(d)(t+0.004+n\cdot 0.01)\cdot\\ &\quad\{\textstyle\sum_{n'=0}^{\infty}\tau(t+0.006+(n+n')\cdot 0.01)\cdot\\ &\quad\ \tau_I(X_0(1-b,(t+0.007+(n+n')\cdot 0.01))\}\end{aligned}$$

$$\begin{aligned}\tau_I(X_0(b,t_0)) &\stackrel{*}{=} \textstyle\int_{t>t_0}\sum_{d\in D} r_1(d)(t)\cdot\\ &\quad\{\ \textstyle\sum_{n=0}^{\infty}\tau(t+0.002+n\cdot 0.01)\cdot s_2(d)(t+0.004+n\cdot 0.01)\cdot\\ &\quad\ \{\textstyle\sum_{n'=0}^{\infty}\tau(t+0.006+(n+n')\cdot 0.01)\cdot\\ &\quad\ \tau_I(X_0(1-b,(t+0.007+(n+n')\cdot 0.01))\}\ \}\end{aligned}$$

The atomic actions occurring in $\tau_I(X_0(b,t_0))$ are independent of the parameter b, thus with RSP we get:

$$PAR_{impl}(t)=\tau_I(X_0(1,t))=\tau_I(X_0(0,t))$$

The expression on the right hand side of the $\stackrel{*}{=}$ can be considered as the specification of the PAR protocol. However, one can say that it contains too much time information and one would expect a specification which states that sooner or later the incoming datum will be send out at port 2. Hence, one needs a mechanism to abstract from some time information. Moreover, one needs to express a notion of fairness, saying that the datum and the acknowledgement can be lost only a finite amount of times. In the above expression this would mean that the n and n' are finite. One very rough way to obtain

this, is to throw away all time information obtaining the (untimed) term:

$$S(b) = \sum_{d \in D} r_1(d) \cdot s_2(d) \cdot S(1-b)$$

9 Conclusions and Further Research

In this report a notion of abstraction is introduced. The adjustment of the model is quite simple and requires the introduction of so-called closure rules. As equivalence we still can use the standard notion of strong bisimulation. The resulting equivalence can be characterized by only one additional τ-law for the calculus without integration.

By interpreting symbolic processes as a special class of timed processes, we obtain a notion of τ equivalence for symbolic processes. The resulting equivalence is delay bisimulation with a strongly rootedness condition.

We can verify a protocol using these laws. To deal with recursion, we need as well the common requirement that every guarded recursive specification has a unique solution (RDP and RSP) and a new principle which we call the *Unwind Principle* (UP).

However, some questions are left open. The completeness proof is given for terms without integration. It is to be expected that the addition of integrals complicates the proof only in a technical way, also because the τ can only be removed if the bounded variable of the associated integral is not used afterwards. But there is a need for techniques dealing with terms with integrals and their transition systems such that these proofs can be done more easily. The statement of the principles RDP, RSP and UP is rather ad hoc and needs further research.

Acknowledgements

The author would like to thank Jos Baeten (Eindhoven Univ. of Technology) for his encouraging comments. Part of the research of this paper was suggested by him. Willem Jan Fokkink (CWI) is thanked for his stylistic advices.

References

[Bae90] J.C.M. Baeten, editor. *Applications of Process Algebra.* Cambridge Tracts in Theoretical Computer Science 17. Cambridge University Press, 1990.

[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.

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

[dR89] W.P. de Roever. Foundations of computer science: Leaving the ivory tower. *Bulletin of the European Association for Theoretical Computer Science*, 44:455–492, 1989.

[Gla87] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings STACS 87*, volume 247 of *Lecture Notes in Computer Science*, pages 336–347. Springer-Verlag, 1987.

[Gro89] J.F. Groote. Transition system specifications with negative premises. Report CS-R8950, CWI, Amsterdam, 1989. An extended abstract appeared in J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90*, Amsterdam, LNCS 458, pages 332–341. Springer-Verlag, 1990.

[GW89] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[HR90] M. Hennessy and T. Regan. A temporal process algebra. Report 2/90, Computer Science Department, University of Sussex, 1990.

[Klu91] A.S. Klusener. Completeness in realtime process algebra. Report CS-R9106, CWI, Amsterdam, 1991. An extended abstract appeared in J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91*, Amsterdam, LNCS 527 , pages 376–392. Springer-Verlag, 1991.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall International, 1989.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 1990.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Ree89] M. Reed. A hierarchy of domains for real-time distributed computing. In *Mathematical Foundations of Programming Language Semantics.* Springer-Verlag, 1989.

[RR88] M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[Wan90] Y. Wang. Real time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer-Verlag, 1990.