Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.C.M. Baeten, J.A. Bergstra

Real time process algebra

Computer Science/Department of Software Technology Report CS-R9053 October

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Real Time Process Algebra

J.C.M. Baeten
Dept. of Software Technology, Centre for Mathematics and Computer Science, P.O.Box 4079, 1009 AB Amsterdam, The Netherlands

J.A. Bergstra
Programming Research Group, University of Amsterdam, P.O.Box 41882, 1009 DB Amsterdam, The Netherlands
Dept. of Philosophy, State University of Utrecht, Heidelberglaan 2, 3584 CS Utrecht, The Netherlands

We describe an axiom system ACPρ that incorporates real timed actions. Many examples are provided in order to explain the intuitive contents of the notation. ACPρ is a generalization of ACP. This implies that some of the axioms have to be relaxed and that ACP can be recovered as a special case from it. The purpose of ACPρ is to serve as a specification language for real time systems. The axioms of ACPρ explain its operational meaning in an algebraic form.

1980 Mathematics Subject Classification (1985 revision): 68Q10, 68Q55, 68Q45, 68Q40.
1987 CR Categories: F.1.2, F.3.2, D.1.3, D.3.1, D.4.1.
Key words & Phrases: real time, process algebra, ACP.
Note: This work is partially sponsored by ESPRIT Basic Research Action 3006, CONCUR, and by RACE project 1046, SPECS. However, this document does not necessarily reflect the views of the SPECS consortium. To appear in Formal Aspects of Computing.

1. INTRODUCTION

The aim of this paper is to explore the possibilities to incorporate real time aspects in process algebra, and in particular in the framework of ACP [BK84]. Many papers have been written about real time processes and their description in various formats that had previously been introduced without real time constraints. We mention [DS89], [GB87], [JM86], [KSRGA88], [MT90], [NRSV90], [R89], [RR86], [RR87], [ZL87]. Our effort is not primarily aimed at a better understanding of real time phenomena but rather more focussed on the question to what extent real time aspects can be incorporated in the setting of ACP. Adding real time features can be done in many ways and it is impossible to explore all options in a single document (see also [G90]). But taking the philosophy of ACP as a starting point several conclusions can be drawn beforehand:

i. Atomic actions take no time, moreover they either exclude one another in time or happen at the same time and are engaged in some kind of communication. This can be preserved if we add to atomic actions a time stamp (as was done in [GV87]). Thus $a(7)$ stands for: perform action a at time 7.

ii. ACP is an executable formalism. In fact its expressions are just programs in an applicative notation. This character must be preserved in the case of real time extensions.

iii. The axioms of ACP are wrong for a real time interpretation: suppose that actions a, b and c cannot communicate. What is $P = (a(2) \cdot c(4)) \parallel b(3)$? According to ACP it is as follows:

Report CS-R9053
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

$(a(2) \cdot c(4)) \between b(3) = (a(2) \cdot c(4)) \lefttriangle b(3) + b(3) \lefttriangle (a(2) \cdot c(4)) =$
$a(2) \cdot (c(4) \between b(3)) + b(3) \cdot a(2) \cdot c(4) =$
$a(2) \cdot (c(4) \cdot b(3) + b(3) \cdot c(4)) + b(3) \cdot a(2) \cdot c(4).$

This is not satisfactory, however, because the intuitive meaning of the entirely deterministic process P is $a(2) \cdot b(3) \cdot c(4)$. In the case of real time processes there is much less room for arbitrary interleaving. In any case it is not always true that parallel composition leads to a combinatorial explosion of execution paths.

iv. After all actions have been provided with time stamps the original system ACP becomes difficult to understand. What exactly is an atomic action without time stamp? Will it obtain a time stamp by being executed? Or is there some other connection? It follows that after the introduction of ACPρ the original system ACP is not just recovered by taking stamped actions instead of ACP's atomic actions. In fact the following can be said:

- ACPρ is a generalization of ACP. It has weaker axioms and the axioms of ACP cannot be maintained for real time processes in general.
- ACP's axioms hold for a subclass of the real time processes. This subclass consists of the so-called *symbolic* processes. An atomic action of a symbolic process is a composed object in terms of ACPρ. Intuitively the symbolic processes uses 'atomic actions' of the form $\underline{a}$ where $\underline{a}$ stands for: 'choose some $a(t)$ for $t \geq 0$'.
- several auxiliary operators are needed to define ACPρ. In the subcase of symbolic processes their occurrences are trivial (superfluous).

v. suppose we have in mind the process $A[2, 3]$ that can perform action a once and can do such at any time t in the interval $[2, 3]$. For this process it is an option to perform the action $a(2.55)$, therefore we expect to find the identity $A[2,3] = A[2,3] + a(2.55)$. Similarly we expect the identity

$$A[2, 3] = A[2, 2.4] + A[2.4, 3].$$

Thus, the choice of a moment in time is to be described by means of + just as all manifestations of choice are. (Recall that a distinction between ACP/CCS on the one hand and CSP on the other hand is that in the first theories only one choice mechanism is provided. This form of simplicity should be preserved.)

vi. We need additional terminology to distinguish between real time process algebra and the form of process algebra not involving time stamps. We propose to use the term 'process algebra' for both 'activities' or 'versions of the theory'. The old case is the denoted with 'symbolic process algebra', the new case with '(real time) process algebra'. Thus the default meaning of 'process algebra' involves the use of time according to this proposal concerning terminology.

The major mechanism of ACP is its description of the merge of processes using arbitrary interleaving. We find that this makes perfect sense in the context of real time processes. There seems to be no immediate need to incorporate notions concerning causality (or so-called 'true concurrency') if a step to real time is made.

The part of the work that becomes harder if not unattractive and difficult is the theory of bisimulations. In principle the notion of a bisimulation generalizes to a real time setting without much difficulty. The problem is that transition systems become very large so that they become hard to visualize. Nev-

ertheless, transition systems are very useful for didactic purposes. On the other hand, it seems that working with the axioms is needed if simple and convincing calculations on processes are to be made. Further, for some recursion equations a solution can only be found by introducing an operational semantics. And then bisimulation is needed to understand that the transition system indeed yields a solution of the recursion equations. (This is needed if one has to deal with processes of the 'Achilles & tortoise' type, i.e. processes that can perform infinitely many consecutive actions in a finite amount of time. In simpler cases one may use a projective limit construction to find solutions of recursion equations.)

The conclusion that we have drawn from this experiment is that it is possible indeed to find a real time version of ACP and that our proposal has the properties that one would like. Before writing this document we have tried several alternatives. Indeed there are even versions of the theory conceivable in which all axioms of ACP remain valid, but unfortunately the meta-theory of those proposals turned out to be quite unsatisfactory. That has led us to the conclusion that ACP has to be generalized in order to accommodate real time, which simply means that at least one of its axioms must be given up. The decision has been made to weaken the axioms concerning the interaction of atomic actions and left merge. The reason to do so is because this turns out to be intuitively plausible. It cannot be excluded at present that quite different generalizations work as well or better.

We make only a few remarks about a theory of abstraction in the real time context, this is to a large extent a matter for future research. Thus it is *concrete* process algebra, the theory without τ's, that is generalized to a real time setting here. Concrete process algebra serves for process specification rather than for verification.

The aims of this research are to propose a language and axioms for the description of real time processes, not to analyse the mathematical foundations of the system. Indeed we run the risk of having an inconsistent set of axioms. The primary method to avoid this is to base ourselves on clear intuitions about the modeling of real time systems. All operators have a clear (informal) operational meaning and the axioms reflect this operational meaning. The purpose of the axioms is to allow an elimination result: all finite closed process expressions can be written in a form using sum and multiplication only. Not even this fact is provided with a rigorous proof however. So this paper does little more than proposing the design of a potential theory for real time processes. It should be noticed however that the major risks for the theory being wrong or useless are not so much in the mathematical foundations but rather in the 'human factors' of the syntax, the axioms and the operational intuitions. In order to support our claim that our language is a workable one indeed we have to propose plausible methods for modeling many real time mechanisms in first.

2. BASIC PROCESS ALGEBRA WITH TIME STAMPS.

First we briefly review the theory BPA (Basic Process Algebra) of [BK84]. Process algebra starts from a given *action alphabet* A (usually finite). Elements of A are called *atomic actions*. BPA has two binary operators: + stands for alternative composition (choice), and · for sequential composition. BPA has the axioms from table 1.

$X + Y = Y + X$	A1
$(X + Y) + Z = X + (Y + Z)$	A2
$X + X = X$	A3
$(X + Y) \cdot Z = X \cdot Z + Y \cdot Z$	A4
$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$	A5

Table 1. BPA.

If we add to BPA a special constant δ standing for *inaction* (comparable to NIL of CCS), we obtain the theory BPAδ. The two axioms for δ are in table 2.

$X + \delta = X$	A6
$\delta \cdot X = \delta$	A7

Table 2. BPAδ = BPA + A6, A7.

When we add real time to this setting, our basic actions are not from the set $A_\delta = A \cup \{\delta\}$, but from the set

$$AT = \{a(t) \mid a \in A_\delta, t \in \mathbb{R}^{\geq 0}\}.$$

Thus, each atomic action is parametrized by a non-negative real number. We use a dense and continuous time domain since that is usual practice in physics (there, sometimes *all* reals are used, as we do in [BB90]; in this case, we need to remove axiom ATA1). Other time domains can also be used, and most of the theory we develop will be unaffected.

Again, the timed actions can be combined by $+,\cdot$. Moreover, we need an additional operation $\gg$, the *(absolute) time shift*. $t \gg X$ denotes the process X starting at time t. This means that all actions that have to be performed at or before time t are turned into deadlocks because their execution has been delayed too long. Clearly $0 \gg X = X$ for all processes X. This is not taken as an axiom because it can be derived for all closed process expressions from the given axioms.

The process $\delta(t)$ can wait until time t, and then, no further activity is possible. The process $\delta(0)$ can perform no activity at all. It obeys the same laws as the process δ of BPAδ, so we can put $\delta = \delta(0)$.

$a(0) = \delta(0)$	ATA1
$\delta(t) \cdot X = \delta(t)$	ATA2
$t < r \Rightarrow \delta(t) + \delta(r) = \delta(r)$	ATA3
$a(t) + \delta(t) = a(t)$	ATA4
$a(t) \cdot X = a(t) \cdot (t \gg X)$	ATA5
$t < r \Rightarrow t \gg a(r) = a(r)$	ATB1
$t \geq r \Rightarrow t \gg a(r) = \delta(t)$	ATB2
$t \gg (X + Y) = (t \gg X) + (t \gg Y)$	ATB3
$t \gg (X \cdot Y) = (t \gg X) \cdot Y$	ATB4

Table 3. BPA$\rho\delta$ = BPAδ + ATA1-5 + ATB1-4.

EXAMPLES:

i. $a(2)\cdot b(3) + \delta(1.5) = a(2)\cdot b(3) + \delta(1.5)\cdot b(3) = (a(2) + \delta(1.5))\cdot b(3) =$
$= (a(2) + \delta(2) + \delta(1.5))\cdot b(3) = (a(2) + \delta(2))\cdot b(3) = a(2)\cdot b(3);$

ii. $a(2)\cdot(b(2)\cdot c(3) + c(1)\cdot c(4) + c(3)\cdot c(2)) =$
$= a(2)\cdot(2 \gg b(2)\cdot c(3) + 2 \gg c(1)\cdot c(4) + 2 \gg c(3)\cdot c(2)) =$
$= a(2)\cdot(\delta(2)\cdot c(3) + \delta(2)\cdot c(4) + c(3)\cdot c(2)) = a(2)\cdot(\delta(2) + \delta(2) + c(3)\cdot c(2)) =$
$= a(2)\cdot(\delta(2) + (\delta(3) + c(3))(3 \gg c(2))) = a(2)\cdot(\delta(3) + c(3))\cdot\delta(3) = a(2)\cdot c(3)\cdot\delta(3).$

A *closed process expression* over $BPA\rho\delta(A)$ is an expression that does not contain process variables or variables for real numbers. We allow every real number as a constant, which means there are uncountably many such closed process expressions. For finite closed process expressions an initial algebra can be defined. This is the initial algebra model of $BPA\rho\delta(A)$. This structure identifies two closed expressions whenever these can be shown identical by means of application of the axioms. We will look at two other models in the following two sections. In these models, recursion equations can be solved.

3. PROJECTIVE LIMIT MODEL.

We will define a projective limit model for $BPA\rho\delta$. This model is useful for the solution of a large class of recursion equations.

Let A be an action alphabet. It may be infinite, since finiteness conditions will have to be imposed on terms rather than on the syntactic categories anyway. Let $PRf(A)$ denote the finite process expressions over action alphabet A with real timed actions. We will work in $PRf = PRf(A)$.

3.1 PROJECTION.

π_t: $PRf \to PRf$ is a mapping defined by the axioms in table 4.

$t \geq t'$	$\Rightarrow$	$\pi_t(a(t')) = a(t')$
$t < t'$	$\Rightarrow$	$\pi_t(a(t')) = \delta(t)$
$\pi_t(X \cdot Y) = \pi_t(X) \cdot \pi_t(Y)$		
$\pi_t(X + Y) = \pi_t(X) + \pi_t(Y)$		

Table 4. Projection.

A *projective sequence* is a mapping p: $[0, \infty) \to PRf$ such that for each pair $t < t'$ of times in $[0, \infty)$, $p(t) = \pi_t(p(t'))$. This leads to a projective limit model for basic process algebra with a real time feature, similar to the model in REED & ROSCOE [RR86], see also [R89], [RR87], [DS89]. An important difference between our approach and theirs is, that in their setting, there is a fixed delay δ following every atomic action.

The elements of our domain are the projective sequences; two sequences p,q are considered to be the same if for all $t \geq 0$ the equation $p(t) = q(t)$ holds in the initial algebra model. The operators are defined component-wise, so e.g. $(p + q)(t) = p(t) + q(t)$.

In this projective limit model it is possible to define the solution of recursion equations. For these equations in most cases one will need processes that are parametrized by real numbers.

3.2 EXAMPLES.

$P_1(t) = a(t) \cdot P_1(t + 1)$

$P_2(t) = a(t) \cdot a(t + 1) \cdot P_2(t + 1)$

$P_3(t) = a(t) \cdot (P_3(t + 1) + c(t + 4)) + b(t + 2) \cdot P_3(t + 4)$

Now we can ask whether or not it is the case that guarded timed equations have unique solutions in the projective limit model. We find that not all of them do. For instance consider the following recursive equation:

$$A(r) = a(2 - r) \cdot A(r/2).$$

In this case $\pi_2(A(1))$ is not a finite process. This is a version of the paradox of Achilles and the tortoise. That paradox seems to be connected to the assumption that reality is a projective limit model in the above sense. It can be concluded that the projective limit model above is not suited for solving recursion equations in general. Therefore, we will look at an operational semantics next.

4. OPERATIONAL SEMANTICS.

First, we will look more closely at the format of process specifications.

4.1 A PROCESS SPECIFICATION FORMAT.

A process specification comprises the following aspects:

i. the description of a signature for a data type, including a sort for the non-negative real numbers denoted by $\mathbb{R}^{\geq 0}$. All constants and operations on the real numbers will have to be named as well;

ii. the description of an algebra $\mathbb{B}$ that serves as a semantics for the data type mentioned in (i). Notice that $\mathbb{B}$ involves a model for the real numbers;

iii. a finite set A of action names with parameter lists. A parameter list is a sequence of sort names from the data type. If $a(S_1,...,S_k)$ is an action name with parameter list then for all data $d_i \in Dom(\mathbb{B},S_i)$ and $t \in \mathbb{R}^{\geq 0}$ $a(d_1,...,d_k)(t)$ is a timed atomic action. (We will also say: a timed and instantiated atomic action). Notice that t is not a parameter of the action in the regular sense, as it plays a special role. On the other hand, one of the S_i may be $\mathbb{R}^{\geq 0}$, in which case time can also be used as a parameter;

iv. a finite collection of process names with parameter lists. As in the case of action names, a parameter list is a possibly empty sequence of sort names. If $P(S_1,...,S_k)$ is a process name with parameter list then for all data $d_i \in Dom(\mathbb{B},S_i)$ $P(d_1, ..., d_k)$ is a so-called instantiated process name. Instantiated process names do not contain a special time parameter, but again, one of the S_i may be $\mathbb{R}^{\geq 0}$;

v. A finite collection of guarded recursion equations in the style of ACP. The meaning of these recursion equations is found by taking all instantiations of the process names that occur in left hand sides. After this instantiation, no free variables may be left in the right-hand sides. Thus a finite system of

equations is viewed as an abbreviation of an infinite collection of recursion equations for instantiated process names.

4.2 DEFINITIONS.

Let a process specification K be given. Then the following matters can be defined:

i. IA(K) denotes the class of instantiated timed actions of K,

ii. IPN(K) denotes the class of instantiated process names for K,

iii. The classes CPE(BPAρδ, K) of closed process expressions made from instantiated timed actions (IA(K)) and instantiated process names (IPN(K)), as introduced in K, combined with the process combinators of BPAρδ. Similarly we will have CPE(ACPρ, K). Further, such classes are found if we add more operators to the syntax of processes. In particular the integral operator will often be used. That leads to notations CPE(BPAρδI, K) and CPE(ACPρI, K).

4.3 STATES.

Given a class L of closed process expressions, say L = CPE(BPAρδ, K) the corresponding class of states is $ST(L) = \{\langle e, t\rangle \mid e \in L \text{ and } t \in \mathbb{R}^{\geq 0}\}$. Thus a state is a closed process expression together with a time. ⟨e, t⟩ denotes the state with time t at which process e still has to be executed.

4.4 ACTION RELATIONS.

Given a process description K, and a class of closed process expressions L over K the operational semantics of K, L consists of three relations

$step \subseteq ST(L) \times IA(K) \times ST(L)$

$idle \subseteq ST(L) \times ST(L)$

$terminate \subseteq ST(L) \times IA(K) \times \mathbb{R}^{\geq 0}$.

The extension of these relations is found as the least fixed point of a simultaneous inductive definition. Before presenting this inductive definition some constraints of the relations will be mentioned. (The inductive definition will guarantee these constraints for its least fixed point.)

step: if step(⟨x, t⟩, a(r), ⟨x', s⟩) then t < r and r = s;

idle: if idle(⟨x, t⟩, ⟨x', r⟩) then t < r and x ≡ x';

terminate: if terminate(⟨x, t⟩, a(r), s) then t < r and r = s.

We write

$\langle x, t\rangle \xrightarrow{a(r)} \langle x', r\rangle$	for	step(⟨x, t⟩, a(r), ⟨x', r⟩),
$\langle x, t\rangle \rightarrow \langle x, r\rangle$	for	idle(⟨x, t⟩, ⟨x, r⟩), and
$\langle x, t\rangle \xrightarrow{a(r)} \langle \surd, r\rangle$	for	terminate(⟨x, t⟩, a(r), r).

We remark that in the step and terminate relations, it is not necessary to mention the time r twice. We prefer to use a somewhat superfluous notation, to make later generalizations possible.

The inductive rules for the operational semantics are similar to those used in structural operational semantics. An exhaustive list is deplorably long. In table 5, we have $a \in IA(K)$, $r,s,t \in \mathbb{R}^{\geq 0}$, $x,y \in L$.

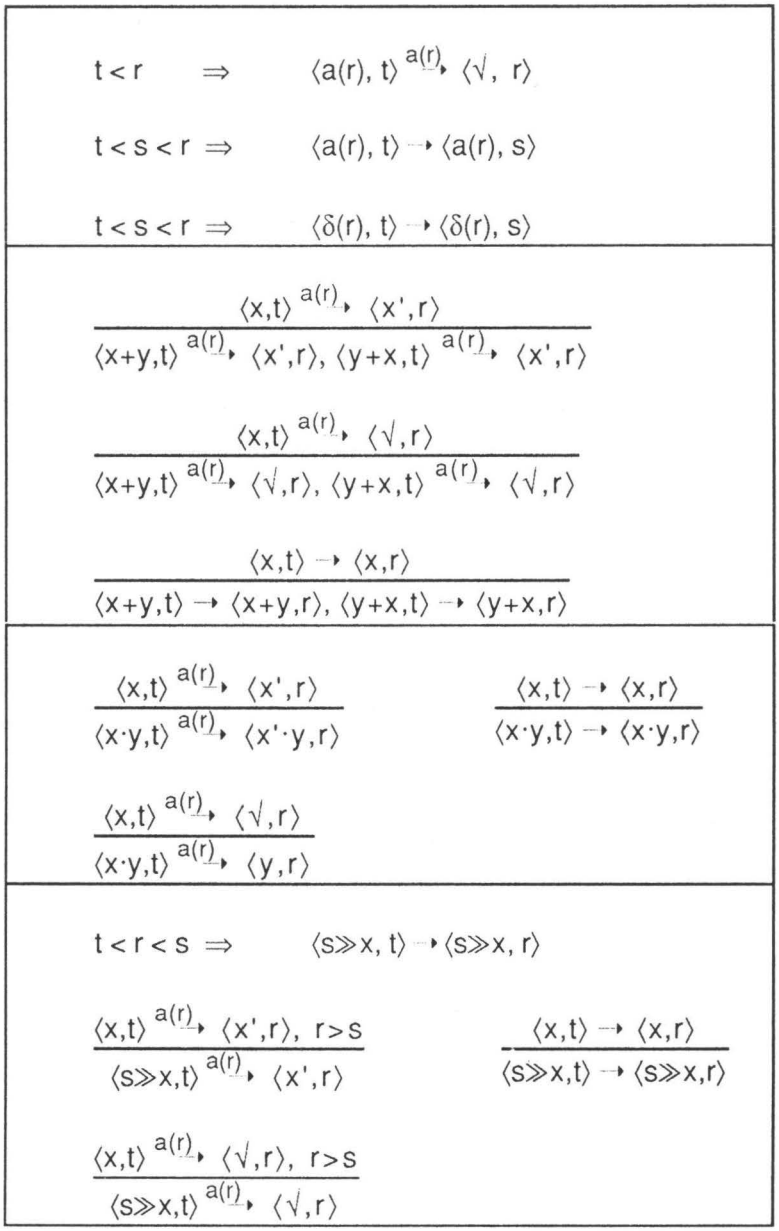

Table 5. Action rules for BPAρδ (inference rules).

Note that the rules for alternative composition express, that by idling, a summand of an alternative composition can be lost if it cannot wait any longer (if the time stamp of an atomic action is reached or passed).

We also need action relation rules for recursive equations. We use recursive equations to specify infinite processes, as in [BK86] or [BBK87], where also the notion of guardedness is explained. Let $X = e$ be a recursive equation, with X a formal variable, and e, e' process expressions, possibly containing the variable X. Then we add the following rules.

$$\frac{\langle e,t\rangle \xrightarrow{a(r)} \langle e',r\rangle}{\langle X,t\rangle \xrightarrow{a(r)} \langle e',r\rangle} \qquad \frac{\langle e,t\rangle \rightarrow \langle e,r\rangle}{\langle X,t\rangle \rightarrow \langle X,r\rangle}$$

$$\frac{\langle e,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle X,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

Table 6. Action rules for recursion.

We let RTS($\langle p, t\rangle$) denote the real time transition system generated from the state $\langle p, t\rangle$ using the structured operational rules above.

4.5 REMARK.

Notice that the inference rules are such that as time progresses without the execution of an action, not more relations become possible, i.e.

$t \le r$ and $\langle x,r\rangle \rightarrow \langle x,s\rangle$ implies $\langle x,t\rangle \rightarrow \langle x,s\rangle$

$t \le r$ and $\langle x,r\rangle \xrightarrow{a(s)} \langle x',s\rangle$ implies $\langle x,t\rangle \xrightarrow{a(s)} \langle x',s\rangle$

$t \le r$ and $\langle x,r\rangle \xrightarrow{a(s)} \langle \surd,s\rangle$ implies $\langle x,t\rangle \xrightarrow{a(s)} \langle \surd,s\rangle$.

These rules do not hold anymore when hidden actions (τ-actions) are present, see section 9.10.

4.6 BISIMULATIONS.

Let a process description K be given using a process syntax (say BPA$\rho\delta$) and consider two process expressions p and q in CPE(BPA$\rho\delta$, K). We will define what it means for the states $\langle p, t\rangle$ and $\langle q, t\rangle$ to be bisimilar.

The cone cone(p, t) of a state over CPE(BPA$\rho\delta$, K) is the collection of all states that are accessible from $\langle p, t\rangle$ in finitely many transitions using the clauses of the structured operational semantics. Then a *bisimulation* between $\langle p, t\rangle$ and $\langle q, t\rangle$ is a relation $R \subseteq \text{cone}(p,t) \times \text{cone}(q,t)$ such that

i. $R(\langle p, t\rangle, \langle q, t\rangle)$;

ii. for each state $\langle p',s\rangle$ in cone(p,t) there is a state $\langle q',s\rangle$ in cone(q,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$;

iii. for each state $\langle q',s\rangle$ in cone(q,t) there is a state $\langle p',s\rangle$ in cone(p,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$;

iv. for each state $\langle p',s\rangle$ in cone(p,t) and state $\langle q',s\rangle$ in cone(q,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is a step a(s') possible from $\langle p',s\rangle$ to state $\langle p'',s'\rangle$, then there is a state $\langle q'',s'\rangle$ in cone(q,t) such that $R(\langle p'',s'\rangle, \langle q'',s'\rangle)$ and there is a step a(s') possible from $\langle q',s\rangle$ to $\langle q'',s'\rangle$.

v. for each state $\langle q',s\rangle$ in cone(q,t) and state $\langle p',s\rangle$ in cone(p,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is a step a(s') possible from $\langle q',s\rangle$ to state $\langle q'',s'\rangle$, then there is a state $\langle p'',s'\rangle$ in cone(p,t) such that $R(\langle p'',s'\rangle, \langle q'',s'\rangle)$ and there is a step a(s') possible from $\langle p',s\rangle$ to $\langle p'',s'\rangle$.

vi. for each state $\langle p',s\rangle$ in cone(p,t) and state $\langle q',s\rangle$ in cone(q,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is idling possible from $\langle p',s\rangle$ to state $\langle p',s'\rangle$, then $R(\langle p',s'\rangle, \langle q',s'\rangle)$ and idling is possible from $\langle q',s\rangle$ to $\langle q',s'\rangle$.

vii. for each state $\langle q',s\rangle$ in cone(q,t) and state $\langle p',s\rangle$ in cone(p,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is idling possible from $\langle q',s\rangle$ to state $\langle q',s'\rangle$, then $R(\langle p',s'\rangle, \langle q',s'\rangle)$ and idling is possible from $\langle p',s\rangle$ to $\langle p',s'\rangle$.

viii. for each state $\langle p',s\rangle$ in cone(p,t) and state $\langle q',s\rangle$ in cone(q,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is a termination step a(s') possible from $\langle p',s\rangle$ to s', then there is a termination step a(s') possible from $\langle q',s\rangle$ to s'.
ix. for each state $\langle q',s\rangle$ in cone(q,t) and state $\langle p',s\rangle$ in cone(p,t) such that $R(\langle p',s\rangle, \langle q',s\rangle)$:
if there is a termination step a(s') possible from $\langle q',s\rangle$ to s', then there is a termination step a(s') possible from $\langle p',s\rangle$ to s'.

We say states $\langle p,t\rangle$ and $\langle q,t\rangle$ are *bisimilar*, notated as $\langle p, t\rangle \leftrightarrow \langle q, t\rangle$, if there exists a bisimulation between $\langle p,t\rangle$ and $\langle q,t\rangle$. Note that conditions ii. and iii. in the definition above are superfluous.

Given a fixed process specification K, then we claim the relation between bisimulations and the axioms of BPAρδ is as follows:

let L(X, Y, Z) = R(X, Y, Z) be an equation of BPAρδ (with variables among X,Y,Z). Then for any process expressions p, q, and r in CPE(BPAρδ, K) and for each $t \in \mathbb{R}^{\geq 0}$ the states

$\langle L(p, q, r), t\rangle$ and $\langle R(p, q, r), t\rangle$ are bisimilar.

It follows that for every fixed instant of time t a model of BPAρδ can be found as follows: map each closed process expression p on the bisimulation class of the state $\langle p, t\rangle$.

We conjecture that in this bisimulation model, all guarded recursive equations have a solution. It would be nice if a proof can be found that all guarded recursive equations even have unique solutions. We leave this second statement as an open problem.

4.7 Some facts on bisimulation.

Later on, we will use the fact that the states $\langle x, t\rangle$ and $\langle t \gg x, t\rangle$ are bisimilar ($\langle x, t\rangle \leftrightarrow \langle t \gg x, t\rangle$) for every closed process expression x and time t. Another useful observation is that $\langle x, t\rangle \leftrightarrow \langle y, t\rangle$ implies that $\langle x, r\rangle \leftrightarrow \langle y, r\rangle$ for every $r \geq t$. In particular, this means that if $\langle x, 0\rangle \leftrightarrow \langle y, 0\rangle$, then $\langle x, t\rangle \leftrightarrow \langle y, t\rangle$ holds for all t.

4.8 Example.

Considering again the 'Achilles and tortoise' process A(1) of 3.2, we see that the cone of $\langle A(1), 0\rangle$ only contains states with time component less than 2, so time 2 cannot be reached, and we have e.g. $\langle A(1), 0\rangle \leftrightarrows \langle A(1)\cdot b(3), 0\rangle$.

5. Integration.

We call *integration* the alternative composition over a continuum of alternatives.

5.1 Suppose that a process notation description is given. This is a process description K without recursion equations. We denote a process notation description with K-. Now it is possible to introduce process expressions with free variables of type $\mathbb{R}^{\geq 0}$. These process expressions can be interpreted as mappings from real numbers to processes. Let Tvar be an infinite collection of variables for $\mathbb{R}^{\geq 0}$. With PE(BPAδ, K-)[Tvar] the collection of process expressions with variables in Tvar is denoted. The notions of free variables and bound occurrence of variables are introduced as usual. Now we add a

binding operator to this term calculus. For each subset V of $\mathbb{R}^{\geq 0}$ and for each variable $v \in$ Tvar and each process expression the following expression is introduced:

$\int_{v \in V} P$, also written as $\mathrm{INT}(v \in V, P)$.

Intuitively this is just the alternative composition of the alternatives P[t/v] for $t \in V$. We use $\int$ rather than Σ to stress the fact that this sum may combine a number of alternatives with the cardinality of the continuum. Of course, v ceases to be a free variable in this expression (if it happened to be one in P). Alpha conversion must be allowed to avoid name clashes.

5.2 OPERATIONAL SEMANTICS.

The operational rules for integration are as follows.

$$\frac{\langle x(u),t\rangle \to \langle x(u),r\rangle,\ u \in V}{\langle \int_{v \in V} x(v),t\rangle \to \langle \int_{v \in V} x(v),r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle x',r\rangle \quad u \in V}{\langle \int_{v \in V} x(v),t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle \quad u \in V}{\langle \int_{v \in V} x(v),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

Table 7. Action relations for integration.

5.3 EXAMPLE

Let $P = \left(\int_{v \in [1,3]} a(v) + \int_{v \in [2,4]} b(v) + \int_{v \in [3,6]} c(v) \right) \cdot d(8)$;

then $\langle P,1.5\rangle \to \langle P,5\rangle$ and $\langle P,2\rangle \xrightarrow{b(3)} \langle d(8),3\rangle$.

5.4 AXIOMS.

The integration operator requires (allows) many additional axioms, see table 8 below.

The last axiom is an extensionality axiom. It is a conditional equation, provided one allows uncountable conjunctions of conditions. Using these axioms a finite process expression involving $\int$ can be written in a normal form in which only prefix multiplication is used (rather than sequential composition in general; see [BK84] for such normal forms).

$\int_{v\in\{t\}} P = P[t/v]$	INT1
$v \notin FV(P) \,\&\, V \neq \emptyset \Rightarrow \int_{v\in V} P = P$	INT2
$\int_{v\in\emptyset} P = \delta$	INT3
$\int_{v\in V\cup W} P = \int_{v\in V} P + \int_{v\in W} P$	INT4
$\int_{v\in V} (P + Q) = \int_{v\in V} P + \int_{v\in V} Q$	INT5
$v \notin FV(Q) \Rightarrow \int_{v\in V} (P\cdot Q) = (\int_{v\in V} P)\cdot Q$	INT6
for all $t \in V$ $(P[t/v] = Q[t/v]) \Rightarrow \int_{v\in V} P = \int_{v\in V} Q$	INT7

Table 8. BPAρδI = BPAρδ + INT1-7.

5.5 EXAMPLE:

$$\int_{v\in[2,3]} \{(a(v+1) + b(2)) \cdot (\int_{v\in[4,6]} b(7-v)) \cdot d(v)\} \cdot c(12) =$$

$$\int_{v\in[2,3]} \{(a(v+1) + b(2)) \cdot (\int_{w\in[4,6]} b(7-w)) \cdot d(v)\} \cdot c(12) =$$

$$\int_{v\in[2,3]} \{(a(v+1) + b(2)) \cdot (\int_{w\in[4,6]} b(7-w) \cdot d(v))\} \cdot c(12) =$$

$$\int_{v\in[2,3]} \{(a(v+1) + b(2)) \cdot (\int_{w\in[4,6]} b(7-w)) \cdot d(v) \cdot c(12)\} =$$

$$\int_{v\in[2,3]} \{a(v+1) \cdot (\int_{w\in[4,6]} b(7-w)) \cdot d(v) \cdot c(12) + b(2) \cdot (\int_{w\in[4,6]} b(7-w)) \cdot d(v) \cdot c(12)\}.$$

5.6 EXAMPLE.

In this example, we will describe three clocks.

i. $C_1(t) = tick(t) \cdot C_1(t+1)$

If one starts the clock in state $\langle C_1(1), 0\rangle$ it will start ticking at time 1 and continue to do so each time unit with absolute precision.

ii. The second clock allows some fluctuations of the ticks.

$$C_2(t) = \int_{v\in[t-0.01,t+0.01]} tick(v) \cdot C_2(t+1)$$

iii. The third clock cumulates the errors:

$$C_3(t) = \int_{v\in[t-0.01,t+0.01]} tick(v)\cdot C_3(v+1).$$

5.7 EXAMPLE.

In this example, we look at processes that describe the automatic switching off of a lamp.

The first example of an automatic switch off mechanism allows to switch on a button at any time; then after 10.5 time units it will be switched off (automatically).

$$B_1 = \int_{v\in\mathbb{R}^{\geq 0}} sw\_on(v)\cdot sw\_off(v+10.5) \cdot B_1$$

An improved version of this device allows to switch on before it has switched off (in order to prevent sudden darkness for instance). We use a parameter t.

$$B_2 = \int_{v \in \mathbb{R}^{\geq 0}} sw\_on(v) \cdot B_2(v+10.5)$$

$$B_2(t) = sw\_off(t) \cdot B_2 + \int_{w \in [0,t)} sw\_on(w) \cdot B_2(w+10.5).$$

A third version of the mechanism allows two different buttons for switching on.

$$B_3 = \int_{v \in \mathbb{R}^{\geq 0}} (sw\_on_1(v) + sw\_on_2(v)) \cdot B_3(v+10.5)$$

$$B_3(t) = sw\_off(t) \cdot B_3 + \int_{w \in [0,t)} (sw\_on_1(w) + sw\_on_2(w)) \cdot B_3(w+10.5).$$

5.8 EXAMPLE.

$$\int_{v \in [3,4]} \delta(v) = \int_{v \in [3,4]} \delta(v) + \int_{v \in \{4\}} \delta(v) = \int_{v \in [3,4]} \delta(v) + \delta(4) = \int_{v \in [3,4]} (\delta(v) + \delta(4)) = \int_{v \in [3,4]} \delta(4) = \delta(4).$$

5.9 ATOMIC ACTIONS WITH REAL VALUED PARAMETERS.

We have introduced the notation $a(b_1, .., b_n)(t)$ for the atomic action a at time t with parameters $b_1, .., b_n$. If some of the b_i may be real numbers, in expressions for them the time t may be used itself. We will provide an example of that in the following.

5.10 EXAMPLE.

Let p be a point in space that travels with uniform velocity v from a to b starting at time t_0. We assume that we work in a one dimensional real space. The distance of a and b is d. Let $u = d/v$ and $t_1 = t_0 + u$. Thus at time t the point is at position $r = a + v \cdot (t - t_0)$. The action $p(r)(t)$ denotes that p signals its presence at r at time t. $p(r)(t)$ will be called a position signal. The process P(t) will describe the traveling point from time t onwards. P(t) allows a position signal at any time and it will terminate after the final position signal $p(b)(t_1)$. The process $p(a)(t_0) \cdot P(t_0)$ describes the traveling point . A recursion equation for P(t) is as follows.

$$P(t) = \int_{t' \in (t,t_1)} p(a + v \cdot (t'-t_0))(t') \cdot P(t') + p(b)(t_1).$$

Notice that this equation may best be considered as a very large system of equations, one for each P(t) with t ranging over the nonnegative real numbers. The advantage of this view is that the P(t) do not have free variables and no complications with name clashes and alpha conversions will arise. Of course it is possible as well to view the recursion equation as a single one but in that case process expressions with parameters have to be understood in detail.

It should be noticed that the process $P(t_0)$ cannot be defined within the projective limit model because there is an infinite sequence of actions possible before time t_1.

6. PARALLEL PROCESSES.

We start off by giving the operational semantics for the parallel composition operator $\|$ (merge). We assume we have given a communication function $| : A_\delta \times A_\delta \rightarrow A_\delta$. $|$ is commutative, associative and δ is a zero element for it.

6.1 OPERATIONAL SEMANTICS.

The operational semantics for merge is easy to understand: in a parallel composition, *both* processes must proceed in order for the composition to proceed. We see 6 possible combinations.

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \rightarrow \langle y,r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle x' \| y,r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y \| x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \rightarrow \langle y,r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

$$\frac{\langle x,t\rangle \rightarrow \langle x,r\rangle,\ \langle y,t\rangle \rightarrow \langle y,r\rangle}{\langle x \| y,t\rangle \rightarrow \langle x \| y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle y',r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x' \| y',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\ \langle y \| x,t\rangle \xrightarrow{c(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle \surd,r\rangle}$$

Table 9. Action rules for parallel composition.

An interesting consequence of these rules is the identity $p \parallel \delta = p \parallel \delta(0) = \delta(0) = \delta$ (a deadlock stops time). Notice the contrast with symbolic process algebra where one can prove for all closed process expressions p that $p \parallel \delta = p \cdot \delta$.

In order to give a axiomatic characterization of parallel composition, we need a number of auxiliary operators.

6.2 ULTIMATE DELAY.

The ultimate delay operator takes a process expression X in $CPE(\Sigma, K)$ for some process signature Σ and a process description K. It returns values in $\mathbb{R}^{\geq 0} \cup \{\omega\}$ with the understanding that $\omega = \sup(\mathbb{R}^{\geq 0})$. The ultimate delay operator U is defined as shown in table 10.

$U(a(t)) = t$	ATU1
$U(\delta(t)) = t$	ATU2
$U(X + Y) = \max\{U(X), U(Y)\}$	ATU3
$U(X \cdot Y) = U(X)$	ATU4

Table 10. $BPA\rho\delta U = BPA\rho\delta$ + ATU1-4.

In order to calculate the ultimate delay of an integral expression, we need an additional axiom.

$U(\int_{v \in V} X) = \sup(\{U(X) \mid v \in V\})$	ATUI

Table 11. BPAρδIU = BPAρδU + BPAρδI + ATUI.

Note that the following interesting equality is a generalization of the law A6 of BPAδ; it holds in the bisimulation model:

$$U(X) \geq t \Rightarrow X = X + \delta(t).$$

Using this identity and the axiom ATUI above, we can derive $\int_{t \in [3,4)} \delta(t) = \delta(4)$ (compare with 5.8).

6.3 BOUNDED INITIALIZATION.

The bounded initialization operator is also denoted by $\gg$, and is the counterpart of the operator with the same name that we saw in the axiomatization of BPAρδ. With $X \gg t$ we denote the process X with its behaviour restricted to the extent that its first action must be performed before time t. Axioms defining $\gg$ are in table 12. Notice that in all axioms a may also be δ.

$t \geq r \Rightarrow a(t) \gg r = \delta(r)$	ATB5
$t < r \Rightarrow a(t) \gg r = a(t)$	ATB6
$(X + Y) \gg t = (X \gg t) + (Y \gg t)$	ATB7
$(X \cdot Y) \gg t = (X \gg t) \cdot Y$	ATB8

Table 12. BPAρδ$\gg$ = BPAρδ + ATB5-8.

In order to calculate the bounded initialization of an integral expression, we need an additional axiom. For this axiom it is assumed that if t happens to be a real number expression it will not contain the free variable v.

$(\int_{v \in V} X) \gg t = \int_{v \in V} (X \gg t)$	ATBI

Table 13. BPAρδI$\gg$ = BPAρδI + BPAρδ$\gg$ + ATBI.

Notice that $X \gg \omega = X$.

6.4 ALGEBRA OF COMMUNICATING PROCESSES.

Now we have defined all auxiliary operators needed for the axiomatization of parallel composition. Let H be some subset of A, and let a,b,c be elements of $A \cup \{\delta\}$.

$a \mid b = b \mid a$	C1
$a \mid (b \mid c) = (a \mid b) \mid c$	C2
$\delta \mid a = \delta$	C3
$t \neq r \Rightarrow a(t) \mid b(r) = \delta(\min(t,r))$	ATC1
$a(t) \mid b(t) = (a \mid b)(t)$	ATC2
$X \parallel Y = X \between Y + Y \between X + X \mid Y$	CM1
$a(t) \between X = (a(t) \gg U(X)) \cdot X$	ATCM2
$(a(t) \cdot X) \between Y = (a(t) \gg U(Y)) \cdot (X \parallel Y)$	ATCM3
$(X + Y) \between Z = X \between Z + Y \between Z$	CM4
$(a(t) \cdot X) \mid b(r) = (a(t) \mid b(r)) \cdot X$	CM5
$a(t) \mid (b(r) \cdot X) = (a(t) \mid b(r)) \cdot X$	CM6
$(a(t) \cdot X) \mid (b(r) \cdot Y) = (a(t) \mid b(r)) \cdot (X \parallel Y)$	CM7
$(X + Y) \mid Z = X \mid Z + Y \mid Z$	CM8
$X \mid (Y + Z) = X \mid Y + X \mid Z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(a(t)) = (\partial_H(a))(t)$	ATD
$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$	D3
$\partial_H(X \cdot Y) = \partial_H(X) \cdot \partial_H(Y)$	D4

Table 14. $ACP\rho = BPA\rho\delta U + BPA\rho\delta\gg +$
$+ C1\text{-}3 + ATC1,2 + CM1,4\text{-}9 + ATCM2,3 + D1\text{-}4 + ATD.$

6.5 REMARK.

Observe that we have the following identity:

$$(a(t) \gg r) \cdot X = (a(t) \gg r) \cdot (t \gg X).$$

PROOF: We have two cases: if $t>r$ then

$$(a(t) \gg r) \cdot X = a(t) \cdot X = a(t) \cdot (t \gg X) = (a(t) \gg r) \cdot (t \gg X);$$

if on the other hand $t \leq r$, then

$$(a(t) \gg r) \cdot X = \delta(r) \cdot X = \delta(r) = \delta(r) \cdot (t \gg X) = (a(t) \gg r) \cdot (t \gg X).$$

6.6 ALTERNATIVE AXIOMS.

As a consequence of this remark, we see that axiom ATCM2 can be phrased as follows:

$$a(t) \between X = (a(t) \gg U(X)) \cdot (t \gg X) \qquad \text{ATCM2'}$$

This version will be used later on (in 7.13).

Also for ATCM3 we will need another version. We can prove that for all closed terms X and Y, axiom ATCM3 is equivalent to the following axiom:

$$(a(t) \cdot X) \between Y = (a(t) \gg U(Y)) \cdot (X \parallel (t \gg Y)) \qquad \text{ATCM3'}.$$

6.7 ELIMINATION THEOREM.

For finite closed process expressions over ACPρ we can formulate the following elimination theorem : Let p be a finite closed process expression over ACPρ. Then there is a finite closed process expression q, not containing the operators $\|$, $\lfloor\!\lfloor$, $\gg$, $\ll$, $\cup$, ∂_H, and with $|$ only on atomic actions (in the form $(a \mid b)(t)$), such that ACPρ $\vdash p = q$. (Notice that q is a BPAρδ-term.)

This theorem makes it possible to prove facts about ACPρ-terms by structural induction, considering only a few cases. The proof of the theorem proceeds by term rewrite analysis (omitted here).

6.8 SYMBOLIC PROCESSES.

Notice that in this setting the atomic action a itself is not a process expression. Only after adding a time stamp t one obtains a process expression $a(t)$. However it is possible to view the original atoms as processes as well. In order to avoid confusion we will underline such occurrences of the atomic action names. We write:

$$\underline{a} = \int_{v \in \mathbb{R}^{\geq 0}} a(v).$$

The processes that are built from these underlined atomic actions constitute a subclass of the space of all processes. As we have mentioned in the introduction we propose to call these processes symbolic processes. (Thus ACP, CCS TCSP etc. are theories about symbolic processes in this terminology.) For symbolic processes one recovers all laws of ACP that have been affected by the introduction of timed actions (see table 15 below).

So we find that as such the axioms of ACP are not a subset of ACPρ. That simply means that ACPρ is a generalization of ACP rather than an extension of it. If one considers symbolic processes only and replaces an atomic action a by its real time version $\underline{a}$ all axioms of ACP are recovered in the following sense: for symbolic processes the axioms of ACP hold in the bisimulation model of section 4. This bisimulation model is a standard model of ACPρ (once we have given action relations for the auxiliary operators, see 6.10).

$X + \underline{\delta} = X$
$\underline{\delta} \cdot X = \underline{\delta}$
$\underline{a} \mid \underline{b} = \underline{b} \mid \underline{a}$
$\underline{a} \mid (\underline{b} \mid \underline{c}) = (\underline{a} \mid \underline{b}) \mid \underline{c}$
$\underline{\delta} \mid \underline{a} = \underline{\delta}$
$\underline{a} \lfloor\!\lfloor X = \underline{a} \cdot X$
$(\underline{a} \cdot X) \lfloor\!\lfloor Y = \underline{a} \cdot (X \| Y)$
$\partial_H(\underline{a}) = \underline{a}$ if $a \notin H$
$\partial_H(\underline{a}) = \underline{\delta}$ if $a \in H$

Table 15. Equations valid for symbolic processes.

Notice that $\underline{\delta}$ is in fact an infinite wait instruction. It follows that deadlock and divergence become identified in this setting.

6.9 ACPρ WITH INTEGRATION.

It is not clear, however, how to find formal derivations of the axioms of ACP from ACPρ. The introduction of $\underline{a}$ brings in the integral notation in the case of merge and requires additional axioms for it. The following axioms for integrals are valid and useful:

$v \notin FV(Q) \Rightarrow (\int_{v\in V} P) \between Q = \int_{v\in V} (P \between Q)$	INT8
$v \notin FV(Q) \Rightarrow (\int_{v\in V} P) \mid Q = \int_{v\in V} (P \mid Q)$	INT9
$v \notin FV(P) \Rightarrow P \mid \int_{v\in V} Q = \int_{v\in V} (P \mid Q)$	INT10
$\partial_H(\int_{v\in V} P) = \int_{v\in V} \partial_H(P)$	INT11

Table 16. ACPρI = BPAρδIU + BPAρδI≫ + ACPρ + INT8-11.

Using these axioms we can extend the elimination theorem of 6.7 to all ACPρI-terms (i.e. we can eliminate the operators mentioned in 6.7, *not* the integral operator). We will provide an example of the effect of this theorem. Suppose $b \mid c = d$ and all other communications equal δ.

$$P = \int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5) =$$
$$= \int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5) + \int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v) \mid c(3.5) +$$
$$+ c(3.5) \between \int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v) =$$
$$\int_{v\in[2,3)} (a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5)) + \int_{v\in[2,3)} (a(v+1)\cdot b(3.5)\cdot c(6+v)) \mid c(3.5)) +$$
$$(c(3.5) \gg U(\int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v)) \between \int_{v\in[2,3)} a(v+1)\cdot b(3.5)\cdot c(6+v) = A + B + C.$$

We can consider these summands independently and we will focus on A (the others presenting similar problems).

$$A = \int_{v\in[2,3)} (a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5)) =$$
$$= \int_{v\in[2,2.5)} (a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5)) + \int_{v\in[2.5,3)} (a(v+1)\cdot b(3.5)\cdot c(6+v) \between c(3.5)) =$$
$$= \int_{v\in[2,2.5)} a(v+1) \gg U(c(3.5))\cdot (b(3.5)\cdot c(6+v) \between c(3.5)) + \int_{v\in[2.5,3)} a(v+1) \gg U(c(3.5))\cdot (b(3.5)\cdot c(6+v) \between c(3.5))$$
$$= \int_{v\in[2,2.5)} a(v+1) \gg 3.5\cdot (b(3.5)\cdot c(6+v) \between c(3.5)) + \int_{v\in[2.5,3)} a(v+1) \gg 3.5\cdot (b(3.5)\cdot c(6+v) \between c(3.5)) =$$
$$= \int_{v\in[2,2.5)} a(v+1)\cdot (b(3.5)\cdot c(6+v) \between c(3.5)) + \int_{v\in[2.5,3)} \delta(3.5)\cdot (b(3.5)\cdot c(6+v) \between c(3.5)) =$$
$$= \int_{v\in[2,2.5)} a(v+1)\cdot (\delta(3.5)(c(6+v) \between c(3.5)) + \delta(3.5)\cdot b(3.5)\cdot c(6+v) + b(3.5)\cdot c(6+v) \mid c(3.5)) +$$
$$+ \int_{v\in[2.5,3)} \delta(3.5) = \int_{v\in[2,2.5)} a(v+1)\cdot (b(3.5) \mid c(3.5)\cdot c(6+v)) + \delta(3.5) = \text{(use the remark in 6.2)}$$
$$= \int_{v\in[2,2.5)} a(v+1)\cdot (b \mid c)(3.5)\cdot c(6+v) = \int_{v\in[2,2.5)} a(v+1)\cdot d(3.5)\cdot c(6+v).$$

6.10 OPERATIONAL SEMANTICS.

In order to obtain an operational model for ACPρI, we also need action relations for all additional operators. We give these straightforward definitions in table 17.

$$\frac{\langle x,t\rangle \to \langle x,r\rangle,\ r<s}{\langle x\gg s,t\rangle \to \langle x\gg s,r\rangle} \qquad \frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ r<s}{\langle x\gg s,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ r<s}{\langle x\gg s,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x,r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \between y,t\rangle \to \langle x \between y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \between y,t\rangle \xrightarrow{a(r)} \langle x' \between y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \to \langle y,r\rangle}{\langle x \between y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle y',r\rangle,\ a|b=c\neq\delta}{\langle x \mid y,t\rangle \xrightarrow{c(r)} \langle x' \between y',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a|b=c\neq\delta}{\langle x \mid y,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\ \langle y \mid x,t\rangle \xrightarrow{c(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a|b=c\neq\delta}{\langle x \mid y,t\rangle \xrightarrow{c(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x,r\rangle}{\langle \partial_H(x),t\rangle \to \langle \partial_H(x),r\rangle} \qquad \frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ a\notin H}{\langle \partial_H(x),t\rangle \xrightarrow{a(r)} \langle \partial_H(x'),r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ a\notin H}{\langle \partial_H(x),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

Table 17. Action relations for auxiliary operators of ACPρ.

6.11 RELATING ACP TO ACPρ.

It is useful to make some additional remarks about the relation between ACP and ACPρ or rather ACP and ACPρI. Suppose we introduce the constant ω for time instants denoting 'infinite' time and we require the axioms of table 18.

$\underline{a} = \int_{v \in \mathbb{R}^{\geq 0}} a(v)$	Ω1
$X \gg \omega = X$	Ω2
$\omega \gg X = \underline{\delta}$	Ω3
$a(\omega) = \underline{\delta}$	Ω4
$U(\underline{a}) = \omega$	Ω5

Table 18. Ω = Ω1-5.

These axioms can be consistently added to ACPρI. The following axioms are reasonable as well:

$U(X) = \omega \Rightarrow U(X + Y) = \omega$	Ω6
$U(X) = \omega \,\&\, U(Y) = \omega \Rightarrow U(X \parallel Y) = \omega$	Ω7
$U(X) = \omega \,\&\, U(Y) = \omega \Rightarrow U(X \between Y) = \omega$	Ω8
$U(X) = \omega \,\&\, U(Y) = \omega \Rightarrow U(X \mid Y) = \omega$	Ω9
$U(X) = \omega \Rightarrow U(\partial_H(X)) = \omega$	Ω10

Table 19. ΩACP = Ω1-10.

Now suppose that a model $\mathbb{M}$ for ACPρI + ΩACP is given that contains at least one process p satisfying U(p) = ω. (The bisimulation models for any process notation are of that kind, see section 4.) Let $\mathbb{M}\mathrm{id}$ (processes in $\mathbb{M}$ with indefinite delay) be the sub-algebra of $\mathbb{M}$ consisting of all processes p that satisfy U(p) = ω. We expect $\mathbb{M}\mathrm{id}$ to be an algebra with respect to the operators of ACP. Let ACP($\underline{A}$) be ACP over A with each atom $a \in A$ replaced by $\underline{a}$. Then $\mathbb{M}\mathrm{id}$ is a model of ACP($\underline{A}$). We see $\mathbb{M}\mathrm{id}$ is a substructure of $\mathbb{M}$ that is a model of ACP provided each atom $a \in A$ is interpreted as $\underline{a}$ ($\in \mathbb{M}\mathrm{id}$) and δ as $\underline{\delta}$.

Typically a process of the form a(t) will not be contained in $\mathbb{M}\mathrm{id}$. So $\mathbb{M}\mathrm{id}$ is not generated by indecomposable primitive processes.

6.12 EXAMPLE: (combination of one bit buffers)
The one bit buffer with input port i and output port j has the following specification in symbolic process algebra:

$B^{ij} = (ri(0) \cdot sj(0) + ri(1) \cdot sj(1)) \cdot B^{ij}$.

Here, we use the so-called *read-send format*. I is a collection of port names and D is some set of data. Then A contains actions si(d) (*send* message d at port i), ri(d) (*read* message d at port i), and ci(d) (*communicate* message d at port i). On these actions, we define the communication function as follows: $si(d) \mid ri(d)) = ci(d)$ (for $d \in D$). In all other cases | yields δ.

We consider several real time versions of the one bit buffer. We assume that time is measured in seconds.

$$Ba^{1,2} = \int_{v \in \mathbb{R}^{\geq 0}} [r1(0)(v) \cdot s2(0)(v+0.01) + r1(1)(v) \cdot s2(1)(v+0.01)] \cdot Ba^{1,2}$$

$$Bb^{1,2}(t) = \int_{v \in [t,\omega)} [r1(0)(v) \cdot s2(0)(v+0.01) + r1(1)(v) \cdot s2(1)(v+0.01)] \cdot Bb^{1,2}(v+0.02)$$

$$Bc^{1,2} = \int_{v\in\mathbb{R}^{\geq 0}} [r1(0)(v)\cdot \int_{w\in[0.009,0.011]} s2(0)(v+w) + r1(1)(v)\cdot \int_{w\in[0.009,0.011]} s2(1)(v+w)] \cdot Bc^{1,2}$$

$$Bd^{1,2}(t) = \int_{v\in[t,\omega)} [r1(0)(v)\cdot \int_{w\in[0.009,0.011]} s2(0)(v+w) + r1(1)(v)\cdot \int_{w\in[0.009,0.011]} s2(1)(v+w)] \cdot Bd^{1,2}(v + 0.02)$$

$$Be^{1,2}(t) = \int_{v\in[t,\omega)} r1(0)(v)\cdot \int_{w\in[0.009,0.011]} s2(0)(v+w)\cdot Be^{1,2}(v + w + 0.01) + r1(1)(v)\cdot \int_{w\in[0.009,0.011]} s2(1)(v+w)\cdot Be^{1,2}(v + w + 0.02).$$

The next step is to consider the parallel composition of several one bit buffers. For instance let the set H(2) contain all read and send actions at port 2 and let H(2, 3, 4) contain all read and send actions at ports 2-4. Then the following processes can be imagined (amongst many others).

$P = \partial_{H(2)}(Be^{1,2}(0.01) \parallel Be^{2,3}(0.01))$

$Q = \partial_{H(2,3,4)}(Ba^{1,2} \parallel Bb^{2,3}(0) \parallel Bc^{3,4}(0) \parallel Be^{4,5}(0))$.

There is a difficulty with the operational semantics of these processes.

Indeed, we have that the only non-trivial communications are ri(d) | si(d) = ci(d). Then it is impossible for P to receive an input along port 1 and to send an output at port 3 at the same time. On the other hand starting in state $\langle P, 0\rangle$ after the actions r1(0)(0.5), c2(0)(0.5105) and waiting for 0.001 seconds the following state is obtained:

$$\langle \partial_{H(2)}[Be^{1,2}(0.5 + 0.0105 + 0.001) \parallel \int_{w\in[0.009,0.011]} s3(1)(0.5105+w) \cdot Be^{2,3}(0.5105 + w + 0.02)],\ 0.5115\rangle =$$

$$\langle \partial_{H(2)}[Be^{1,2}(0.5115) \parallel \int_{w\in[0.009,0.011]} s3(1)(0.5105+w) \cdot Be^{2,3}(0.5305 + w)],\ 0.5115\rangle.$$

In this state one would expect that the the actions r1(0)(0.52) and s3(1)(0.52) are possible simultaneously, because these actions are entirely independent in the architecture of P. We will get around this difficulty by introducing a new communication function on the atomic actions. This would be possible in ACP just the same, but there it is less needed because it does not contradict intuition so much.

6.13 MULTI-ACTIONS.

We will illustrate the same problem in a much simpler setting. Suppose that a system has two external ports 1 and 2 and various internal ports. To 1 the component P is connected whereas Q is connected to port 2. The programs for P and Q are fairly trivial: both will write a value at time 3 and thereafter terminate. Thus P = s1(7)(3), Q = s2(6)(3). Other components of the system perform no action at all. Now consider the parallel composition.

$P \parallel Q = s1(7)(3) \parallel s2(6)(3) = s1(7)(3) \between s2(6)(3) + s2(6)(3) \between s1(7)(3) + s1(7)(3) \mid s2(6)(3)$

$= (s1(7)(3) \gg U(s2(6)(3))) \cdot s2(6)(3) + (s2(6)(3) \gg U(s1(7)(3))) \cdot s1(7)(3) + s1(7)(3) \mid s2(6)(3)$

$= (s1(7)(3) \gg 3) \cdot s2(6)(3) + (s2(6)(3) \gg 3) \cdot s1(7)(3) + s1(7)(3) \mid s2(6)(3) =$

$= \delta(3) + \delta(3) + (s1(7) \mid s2(6))(3) = (s1(7) \mid s2(6))(3).$

In the standard format of read send communication that is always used in symbolic ACP the communication $s1(7) \mid s2(6)$ will be δ and this implies $P \parallel Q = \delta(3)$. It works perfectly well in the symbolic case: $s1(7) \parallel s2(6) = s1(7) \cdot s2(6) + s2(6) \cdot s1(7)$. So the arbitrary interleaving avoids a deadlock that is found in the real time case. The solution to these difficulties is simple: if actions are along different ports they are independent. That means that a communication leads to a multi-action. In a multi-action only one action per port may be contained. So we write $s1(7) \mid s2(6) = s1(7) \,\&\, s2(6)$ and view the multi-atom $s1(7) \,\&\, s2(6)$ as a new atomic action formally. Then we find: $P \parallel Q = (s1(7) \,\&\, s2(6))(3)$ which corresponds to the intuitions much better.

6.14 COMMUNICATION FUNCTION.

In this section we will describe a new communication function for the read send format with handshaking communication at internal ports, that allows parallel execution of independent actions. This mechanism can be completely described in ACP, so does not involve any real time aspects. Nevertheless, in symbolic process algebra it is not needed because the arbitrary interleaving prevents deadlocks in those cases.

We will extend the original alphabet A. Suppose that we have a set of *locations* L, $A_\delta = A \cup \{\delta\}$, and we have a *location function* $\lambda: A_\delta \to L$ and a *communication function* $\gamma: A_\delta \times A_\delta \to A_\delta$ such that the conditions in table 20 hold.

$\lambda(a) \neq \lambda(b) \Rightarrow \gamma(a, b) = \delta$
$a \neq \delta \Rightarrow \lambda(a) \neq \lambda(\delta)$
$\gamma(a, b) \neq \delta \Rightarrow \lambda(a) = \lambda(b) = \lambda(\gamma(a, b))$
$\gamma(a, b) = \gamma(b, a)$
$\gamma(a, \gamma(b, c)) = \gamma(\gamma(a, b), c)$
$\gamma(a, \delta) = \delta$

Table 20. Conditions on λ, γ.

As an example, we can use the standard read-send format. Then I is the set of labels, and λ,γ are defined as follows:

$\lambda(si(d)) = \lambda(ri(d)) = \lambda(ci(d)) = i$;

$\gamma(si(d), ri(d)) = ci(d)$ for $d \in D$. In all other cases γ yields δ.

Now we extend the set A to the new set of atomic actions A&. A& consists of *sets* of actions of A, e.g. $a_1 \,\&\, a_2 \,\&\, \ldots \,\&\, a_n$, such that all elements have different locations, i.e. $i \neq j \Rightarrow \lambda(a_i) \neq \lambda(a_j)$. The order of the actions does not matter, i.e. & is commutative and associative. We put $A\&_\delta = A\& \cup \{\delta\}$. The elements of $A\&_\delta$ are intended as simultaneous executions of actions of A_δ. We will call the new atoms multi-atoms. Notice that if A_δ is finite then $A\&_\delta$ is a finite superset of it.

Now $A\&_\delta$ is the new alphabet, on the basis of which ACP (resp. ACPρ) is defined. The communication merge $\mid$ is defined on this alphabet as follows: in

$$a_1 \,\&\, a_2 \,\&\, \ldots \,\&\, a_n \mid b_1 \,\&\, b_2 \,\&\, \ldots \,\&\, b_k,$$

we find, for each pair i,j with $\lambda(a_i) = \lambda(b_j)$, the communication $\gamma(a_i, b_j)$, and then we take the conjunction of these communication actions with the remaining single actions. If at least one communication is δ, the whole term becomes δ. Thus, in the read-send format, we have

$$s1(d) \,\&\, r2(e) \mid s2(e) \,\&\, s3(f) = s1(d) \,\&\, c2(e) \,\&\, s3(f), \text{ and}$$
$$s1(d) \,\&\, r2(e) \mid s1(d) \,\&\, s2(e) = \delta.$$

We see that this definition of $\mid$ satisfies the conditions for a communication function (axioms C1-3 of ACPρ, see 6.4). We also get the result desired in 6.13: $s1(7)(3) \parallel s2(6)(3) = (s1(7) \,\&\, s2(6))(3)$.

This construction is better suited in the case of real time process algebra than the definition in which $\mid$ coincides with γ on the atomic actions. Notice however that all of this firmly stays within the setting of ACP.

Instead of talking about locations, we can consider the more general case where atomic actions do not only a time component, but also space components. Then, the case described here is a degenerated case of the classical real space setting. For more details, see [BB90].

We remark that there is also another solution to this problem: instead of giving an exact timing to each action, we can also assign a small interval. Thus, if ε is some small positive number, we do not consider $s1(7)(3) \parallel s2(6)(3)$, but instead $\int_{t\in(3-\varepsilon,3+\varepsilon)} s1(7)(t) \parallel \int_{t\in(3-\varepsilon,3+\varepsilon)} s2(6)(t)$. In the latter process, we get a correct result, when we use interleaving.

6.15 EXAMPLE.

We describe the *Positive Acknowledgement with Retransmission* protocol (see [T81]) that is based on time outs of acknowledgements. We picture the layout of the system in fig. 1. In order to avoid duplicates, messages from a set D are labeled with an alternating bit from $B = \{0,1\}$.

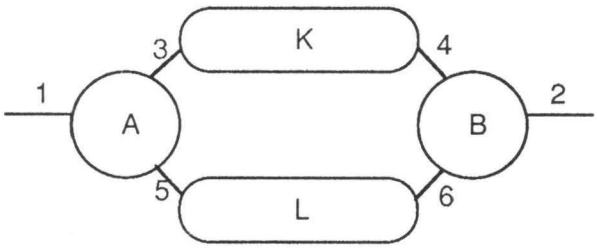

FIGURE 1.

$$A = A(0)$$

$$A(b) = \int_{v\in\mathbb{R}^{\geq 0}} \sum_{d\in D} r1(d)(v) \cdot A(b,d,v)$$

$$A(b,d,v) = s3(db)(v+0.001) \cdot [\int_{w\in[0,v+0.01)} r5(ack)(w)\cdot A(1-b) \;+\; time\_out(v+0.01)\cdot A(b,d,\; v+0.01)]$$

$$K = \int_{v\in\mathbb{R}^{\geq 0}} \sum_{f\in D\times B} r3(f)(v) \cdot [s4(f)(v+0.002) + error_K(v+0.001)] \cdot K$$

$$L = \int_{v\in\mathbb{R}^{\geq 0}} r6(ack)(v) \cdot [s5(ack)(v+0.002) + error_L(v+0.001)] \cdot L$$

$B = B(0)$

$$B(b) = \int_{v\in\mathbb{R}^{\geq 0}} \sum_{d\in D} r4(db)(v) \cdot s2(d)(v+0.001) \cdot B(1\text{-}b,v) + \int_{v\in\mathbb{R}^{\geq 0}} \sum_{d\in D} r4(d(1\text{-}b))(v) \cdot B(b,v)$$

$$B(b,v) = s6(ack)(v+0.002) \cdot B(b).$$

The protocol is then constructed as follows:

$$PAR = \partial_H(A \parallel K \parallel L \parallel B)$$

where H contains all read and send actions along the internal ports 3, 4, 5 and 6. The protocol works correctly as long as the time out time is larger than the time of a complete protocol cycle (here 0.007 seconds). Using real time process algebra, we do not need a priority operator in the description, as in [V90].

7. Relative time.

With a[t] we denote an action a that has to be performed t time units after its enabling occurrence of · has passed. The initialization of a process counts as such a point in time as well. Thus $a[2] \cdot b[1] \cdot c[4]$ denotes: 2 time units after the start (at some moment to be determined by an environment) the action a is performed, 1 time unit later b happens and then 4 time units later c must be planned.

7.1 Basic Process Algebra with relative time.

Now we present the axiom system BPA in the case of relative time. The r in the names of the axiom systems indicates that we deal with relative time.

$a[0] = \delta[0]$	RTA1
$\delta[t] \cdot X = \delta[t]$	RTA2
$t < r \Rightarrow \delta[t] + \delta[r] = \delta[r]$	RTA3
$a[t] + \delta[t] = a[t]$	RTA4

Table 21. $BPA_{r\rho\delta} = BPA_\delta$ + RTA1-4.

7.2 Examples.

i. $a[2] \cdot b[3] + \delta[1.5] = a[2] \cdot b[3] + \delta[1.5] \cdot b[3] = (a[2] + \delta[1.5]) \cdot b[3] =$
$= (a[2] + \delta[2] + \delta[1.5]) \cdot b[3] = (a[2] + \delta[2]) \cdot b[3] = a[2] \cdot b[3]$

ii. $a[2] \cdot (b[2] \cdot c[3] + c[1] \cdot c[4] + \delta[3] \cdot c[2]) = a[2] \cdot (b[2] \cdot c[3] + c[1] \cdot c[4] + \delta[3])$

7.3 Examples of recursion equations.

$P_1 = a[1] \cdot P_1$

$P_2 = a[1] \cdot a[2] \cdot P_2$

$P_3 = a[0.5] \cdot (P_3 + c[3.5]) + b[2]) \cdot P_3$

7.4 DEFINITION

A *state* is a pair $\langle e, t\rangle$ with e a closed process expression and t a time instant. Thus the recursion equations above induce states such as $\langle P_1, 2\rangle$ and $\langle b[2] \cdot (P_1 + P_2) + P_3, 3\rangle$.

7.5 EXAMPLE.

Starting the following process A at time 0 with r initialized at 1

$$A(r) = a[r] \cdot A(r/2)$$

leads to the process $a[1]\cdot a[0.5]\cdot a[0.25]$ ... All actions of this process happen before $t = 2$.

7.6 OPERATIONAL SEMANTICS

Here, we do not specify a projective limit model, since, as in the absolute time case, we want to focus on an operational model. The action rules for the operational model follow.

$$\langle a[r], t\rangle \xrightarrow{a(t+r)} \langle \surd, t+r\rangle$$

$$s < r \Rightarrow \quad \langle a[r], t\rangle \to \langle a[r-s], t+s\rangle$$

$$s < r \Rightarrow \quad \langle \delta[r], t\rangle \to \langle \delta[r-s], t+s\rangle$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x+y,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y+x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x+y,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y+x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x',r\rangle,\ \langle y,t\rangle \to \langle y',r\rangle}{\langle x+y,t\rangle \to \langle x'+y',r\rangle}$$

$$\frac{\langle x,t\rangle \to \langle x',r\rangle,\ t+\tilde{U}(y)\leq r}{\langle x+y,t\rangle \to \langle x',r\rangle,\ \langle y+x,t\rangle \to \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x\cdot y,t\rangle \xrightarrow{a(r)} \langle x'\cdot y,r\rangle} \qquad \frac{\langle x,t\rangle \to \langle x',r\rangle}{\langle x\cdot y,t\rangle \to \langle x'\cdot y,r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x\cdot y,t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

Table 22. Operational semantics of BPArρδ.

7.7 COMMENTS.

When we use relative time, idling may change the process expression we are dealing with, as the axioms in the first block of table 22 show. This complicates the rules for alternative composition, as idling may cause the loss of a summand (when its relative time becomes 0). The operator $\tilde{U}$ is the relative time version of the operator U defined in 6.2:

$\tilde{U}(a[t]) = t$	RTU1
$\tilde{U}(\delta[t]) = t$	RTU2
$\tilde{U}(X + Y) = \max\{\tilde{U}(X), \tilde{U}(Y)\}$	RTU3
$\tilde{U}(X \cdot Y) = \tilde{U}(X)$	RTU4

Table 23. $BPAr\rho\delta\tilde{U} = BPAr\rho\delta$ + RTU1-4.

Similar to 6.2, $BPAr\rho\delta I\tilde{U} = BPAr\rho\delta\tilde{U} + BPAr\rho\delta I$ + RTUI.

Notice that the condition $t+\tilde{U}(y)\leq r$ is equivalent to the following statement:

there is no expression y' such that $\langle y, t\rangle \rightarrow \langle y', r\rangle$.

We do not want to include a negative condition in an operational rule, as that may cause problems (the rules may not define a relation, see GROOTE [G89]).

The definition of bisimulation needs some adaptations, in order to reflect that process expressions may change by idling. These adaptations are straightforward.

7.8 INTEGRATION.

The integration operator requires the same axioms as in 5.4, so we put $BPAr\rho\delta I = BPAr\rho\delta$ + INT1-7. The operational semantics becomes much more complicated.

$$\frac{\text{for all } u\in S\ \langle x(u),t\rangle \rightarrow \langle x'(u),r\rangle,\ \emptyset\neq S\subseteq V,\ \text{and for all } u\in V\text{-}S\ \ t+\tilde{U}(x(u))\leq r}{\langle \int_{v\in V} x(v),t\rangle \rightarrow \langle \int_{v\in S} x'(v),r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ u\in V}{\langle \int_{v\in V} x(v),t\rangle \xrightarrow{a(r)} \langle x',r\rangle}$$

$$\frac{\langle x(u),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ u\in V}{\langle \int_{v\in V} x(v),t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}$$

Table 24. Action relations for integration, relative time.

7.9 EXAMPLES OF THE USE OF INTEGRATION.

i. Three clocks (the same examples as in 5.6).

$Cl_1 = tick[1] \cdot Cl_1$

Notice that here we do not need a parameter for the process, which we did need in 5.6.i. The second clock does need a parameter:

$$Cl_2(w) = \int_{v\in[-0.01,0.01]} tick[1+v-w] \cdot Cl_2(v)$$

Starting this clock at time 0 with $w = 0$ may lead to the following execution:

at $t = 1.004$	$tick[1 + 0.004]$	$Cl_2(0.004)$
at $t = 2.003$	$tick[1 + 0.003 - 0.004]$	$Cl_2(0.003)$
at $t = 2.998$	$tick[1 - 0.002 - 0.003]$	$Cl_2(- 0.002)$
at $t = 4.001$	$tick[1 + 0.001 + 0.002]$	$Cl_2(0.001)$, and so on.

The third clock cumulates the errors:

$$Cl_3 = \int_{v\in[-0.01,0.01]} tick[1+v] \cdot Cl_3.$$

ii. Automatic switch off (see 5.7).

The first version is very similar to 5.7.

$$B_1 = \int_{v\in\mathbb{R}^{\geq 0}} sw\_on[v]\cdot sw\_off[10.5)] \cdot B_1$$

The second version does not need a parameter t.

$$B_2 = \int_{v\in\mathbb{R}^{\geq 0}} sw\_on[v]\cdot B_2^*)$$

$$B_2^* = sw\_off[10.5]\cdot B_2 + \int_{w\in[0,10.5)} sw\_on[w]\cdot B_2^*.$$

7.10 Operational semantics for ACPrρ.

Now we investigate parallel composition in the present setting. The operational semantics is very much like the one given in 6.1 (as in 7.6, a process expression may change by idling).

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \rightarrow \langle y',r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle x' \| y',r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y' \| x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \rightarrow \langle y',r\rangle}{\langle x \| y,t\rangle \xrightarrow{a(r)} \langle y',r\rangle,\ \langle y \| x,t\rangle \xrightarrow{a(r)} \langle y',r\rangle}$$

$$\frac{\langle x,t\rangle \rightarrow \langle x',r\rangle,\ \langle y,t\rangle \rightarrow \langle y',r\rangle}{\langle x \| y,t\rangle \rightarrow \langle x' \| y',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle y',r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x' \| y',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle x',r\rangle,\ \langle y \| x,t\rangle \xrightarrow{c(r)} \langle x',r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle,\ \langle y,t\rangle \xrightarrow{b(r)} \langle \surd,r\rangle,\ a \mid b=c\neq\delta}{\langle x \| y,t\rangle \xrightarrow{c(r)} \langle \surd,r\rangle}$$

Table 25. Action rules for parallel composition with relative time.

7.11 THE BOUNDED INITIALIZATION OPERATOR FOR RELATIVE TIME.

Our next goal is an axiomatization of merge in the present setting. Again, we need a number of auxiliary operators. We have seen the ultimate delay operator for relative time already. We give axioms for the bounded initialization operator for relative time (cf. 6.3), and the time shift operator for relative time (cf. section 2).

First, the bounded initialization operator for relative time, $\tilde{\gg}$.

$t \geq r \Rightarrow a[t] \tilde{\gg} r = \delta[r]$	RTB5
$t < r \Rightarrow a[t] \tilde{\gg} r = a[t]$	RTB6
$(X + Y) \tilde{\gg} t = (X \tilde{\gg} t) + (Y \tilde{\gg} t)$	RTB7
$(X \cdot Y) \tilde{\gg} t = (X \tilde{\gg} t) \cdot Y$	RTB8

Table 26. Bounded initialization.

7.12 THE TIME SHIFT OPERATOR FOR RELATIVE TIME.

Next, we present axioms for the time shift operator for relative time, also denoted by $\tilde{\gg}$.

$t < r \Rightarrow t \tilde{\gg} a[r] = a[r - t]$	RTB1
$t \geq r \Rightarrow t \tilde{\gg} a[r] = \delta[t]$	RTB2
$t \tilde{\gg} (X + Y) = (t \tilde{\gg} X) + (t \tilde{\gg} Y)$	RTB3
$t \tilde{\gg} (X \cdot Y) = (t \tilde{\gg} X) \cdot Y$	RTB4

Table 26. BPArρδ$\tilde{\gg}$ = BPArρδ + RTB1-8.

The *shift operator* represents a process from a later initial time. We see $2 \tilde{\gg} a[5] = a[3]$ and $3 \tilde{\gg} a[5] \cdot b[7] = a[2] \cdot b[7]$.

Again, $BPA_{rp\delta I}\tilde{\gg} = BPA_{rp\delta}\tilde{\gg} + BPA_{rp\delta I} + RTBI$, where RTBI is the obvious analogon of axiom ATBI in 6.3 (replace $\gg$ by $\tilde{\gg}$).

Note that it is straightforward to give an operational semantics of the bounded initialization operator and the time shift operator.

7.13 Algebra of communicating processes for relative time.

Now we are able to introduce a setting with communication. The complication is with interleaving. If one explains a merge in terms of interleaving, the execution of an action in one component of the merge has the effect that in an other component the initialization time is shifted. This effect is described in the equations below. The r in ACP_{rp} indicates that this is an axiom system using relative time only. In order to provide an intuition for the equations, notice that we expect the following identities to be derivable:

$a[2] \parallel (b[3] \cdot c[5]) = a[2] \cdot b[1] \cdot c[5]$

$a[3] \parallel (b[2] \cdot c[5]) = b[2] \cdot a[1] \cdot c[4]$

$a[2] \parallel \delta[1] = \delta[1]$

$a[3] \parallel \delta[5] = a[3] \cdot \delta[2]$.

We find that the solution to this problem is obtained when we use the variants ATCM2' and ATCM3' from 6.6. This gives the axioms RTCM2 and RTCM3.

Notice that also in the case of ACP_{rp} we can put $\underline{a} = \int_{v \in \mathbb{R}^{\geq 0}} a[v]$, and obtain results similar to 6.8.

Let $|$ be a communication function. $|: A_\delta \times A_\delta \to A_\delta$, $|$ is commutative, associative and δ is a zero element for it. H is some subset of A.

$a \mid b = b \mid a$	C1
$a \mid (b \mid c) = (a \mid b) \mid c$	C2
$\delta \mid a = \delta$	C3
$t \neq r \Rightarrow a[t] \mid b[r] = \delta[\min(t,r)]$	RTC1
$a[t] \mid b(t) = (a \mid b)[t]$	RTC2
$X \parallel Y = X \mathbin{\mathbb{L}} Y + Y \mathbin{\mathbb{L}} X + X \mid Y$	CM1
$a[t] \mathbin{\mathbb{L}} X = (a[t] \mathbin{\tilde{\gg}} \tilde{U}(X)) \cdot (t \mathbin{\tilde{\gg}} X)$	RTCM2
$(a[t] \cdot X) \mathbin{\mathbb{L}} Y = (a[t] \mathbin{\tilde{\gg}} \tilde{U}(Y)) \cdot (X \parallel (t \mathbin{\tilde{\gg}} Y))$	RTCM3
$(X + Y) \mathbin{\mathbb{L}} Z = X \mathbin{\mathbb{L}} Z + Y \mathbin{\mathbb{L}} Z$	CM4
$(a[t] \cdot X) \mid b[r] = (a[t] \mid b[r]) \cdot X$	CM5
$a[t] \mid (b[r] \cdot X) = (a[t] \mid b[r]) \cdot X$	CM6
$(a[t] \cdot X) \mid (b[r] \cdot Y) = (a[t] \mid b[r]) \cdot (X \parallel Y)$	CM7
$(X + Y) \mid Z = X \mid Z + Y \mid Z$	CM8
$X \mid (Y + Z) = X \mid Y + X \mid Z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(a[t]) = (\partial_H(a))[t]$	RTD
$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$	D3
$\partial_H(X \cdot Y) = \partial_H(X) \cdot \partial_H(Y)$	D4

Table 28. $ACPr\rho = BPAr\rho\delta U + BPAr\rho\delta\tilde{\gg} +$
$+ C1\text{-}3 + RTC1,2 + CM1,4\text{-}9 + RTCM2,3 + D1\text{-}4 + RTD.$

7.14 EXAMPLE (combination of one bit buffers, see 6.12).
We consider several versions, with relative time. In the second and last case, we can do without a process parameter, in the fourth case we still need it.

$$Ba^{1,2} = \int_{v \in \mathbb{R}^{\geq 0}} \{r1(0)[v] \cdot s2(0)[0.01] + r1(1)[v] \cdot s2(1)[0.01]\} \cdot Ba^{1,2}$$

$$Bb^{1,2} = \int_{v \in [0.01,\omega]} \{r1(0)[v] \cdot s2(0)[0.01] + r1(1)[v] \cdot s2(1)[0.01]\} \cdot Bb^{1,2}$$

$$Bc^{1,2} = \int_{v \in \mathbb{R}^{\geq 0}} \{r1(0)[v] \cdot \int_{w \in [0.009,0.011]} s2(0)[w] + r1(1)[v] \cdot \int_{w \in [0.009,0.011]} s2(1)[w]\} \cdot Bc^{1,2}$$

$$Bd^{1,2}(t) = \int_{v \in [t,\omega]} \{r1(0)[v] \cdot \int_{w \in [0.009,0.011]} s2(0)[w] + r1(1)[v] \cdot \int_{w \in [0.009,0.011]} s2(1)[w]\} \cdot Bd^{1,2}(0.02 - w)$$

$$Be^{1,2} = \int_{v \in [0.01,\omega]} r1(0)[v] \cdot \int_{w \in [0.009,0.011]} s2(0)[w] \cdot Be^{1,2} + r1(1)[v] \cdot \int_{w \in [0.009,0.011]} s2(1)[w] \cdot Be^{1,2*}.$$

$$Be^{1,2*} = \int_{v\in[0.02,\omega]} r1(0)[v]\cdot \int_{w\in[0.009,0.011]} s2(0)[w]\cdot Be^{1,2} + r1(1)[v]\cdot \int_{w\in[0.009,0.011]} s2(1)[w]\cdot Be^{1,2*}.$$

8. COMBINING ABSOLUTE AND RELATIVE TIME.

One may imagine process expressions involving both absolute and relative time constructs. E.g.:

- $a[2] \cdot (b[3] + c[2]) \cdot d(6)$
- $(a[2] \parallel b(2)) \cdot (c[3] \parallel d(3))$.

Such expressions cannot be put in normal form, nor is there a straightforward way to eliminate $\parallel$ or ∂_H. Direct integration of ACPρ and ACPrρ fails because there is no clear way to define the ultimate delay operator U, or bounded initialization and time shift operator $\gg$ on relative time process expressions, such as

$U(a[6]\cdot b[2])$ $a[6] \gg 4$ $2 \gg a[4]$.

Similar difficulties arise with

$\tilde{U}(a(6))$ $a(3) \tilde{\gg} 2$ $3 \tilde{\gg} a(2)$.

8.1 TIME SHIFT OPERATOR.

First of all, we remark that the time shift operator for absolute time does not give us difficulties. This is because the following simple conversion axioms are valid.

$t \gg a[0] = \delta(t)$	ARB1
$r > 0 \Rightarrow t \gg a[r] = a(t + r)$	ARB2
$t \gg (a[r] \cdot X) = (t \gg a[r]) \cdot X$	ARB3

Table 29. Absolute time shift conversion.

8.2 TRANSLATION RULES.

The following rules are also useful, and give back and forth transformations of mixed time expressions, but no normalization or elimination.

$a(t) \cdot b[r] = a(t) \cdot b(t + r)$
$a(t) \cdot b[r] \cdot X = a(t) \cdot b(t + r) \cdot X$
$a(t) \cdot (b[r] + X) = a(t) \cdot (b(t + r) + X)$
$a(t) \cdot (b[r] \cdot X + Y) = a(t) \cdot (b(t + r) \cdot X + Y)$
$a(t) \cdot (b[r] \parallel X) = a(t) \cdot (b(t + r) \parallel X)$
$a(t) \cdot (b[r] \cdot X \parallel Y) = a(t) \cdot (b(t + r) \cdot X \parallel Y)$
$a(t) \cdot \partial_H(b[r] \parallel X) = a(t) \cdot \partial_H(b(t + r) \parallel X)$
$a(t) \cdot \partial_H(b[r] \cdot X \parallel Y) = a(t) \cdot \partial_H(b(t + r) \cdot X \parallel Y)$
$a(t) \cdot (b[r] \between X) = a(t) \cdot (b(t + r) \between X)$
$a(t) \cdot (b[r] \cdot X \between Y) = a(t) \cdot (b(t + r) \cdot X \between Y)$
$a(t) \cdot \partial_H(b[r] \between X) = a(t) \cdot \partial_H(b(t + r) \between X)$
$a(t) \cdot \partial_H(b[r] \cdot X \between Y) = a(t) \cdot \partial_H(b(t + r) \cdot X \between Y)$
$a(t) \cdot (b[r] \mid X) = a(t) \cdot (b(t + r) \mid X)$
$a(t) \cdot (b[r] \cdot X \mid Y) = a(t) \cdot (b(t + r) \cdot X \mid Y)$
$a(t) \cdot \partial_H(b[r] \mid X) = a(t) \cdot \partial_H(b(t + r) \mid X)$
$a(t) \cdot \partial_H(b[r] \cdot X \mid Y) = a(t) \cdot \partial_H(b(t + r) \cdot X \mid Y)$

Table 30. Translation rules.

8.3 REMAINING OPERATORS.

Now we focus on the problems with the ultimate delay, bounded initialization and time shift operators mentioned in the beginning of this section. First of all, notice we have the following reduction axioms.

$U(a[t] \cdot X) = U(a[t])$	ARU1
$\tilde{U}(a(t) \cdot X) = \tilde{U}(a(t))$	ARU2
$(a[t] \cdot X) \gg r = (a[t] \gg r) \cdot X$	ARB4
$(a(t) \cdot X) \tilde{\gg} r = (a(t) \tilde{\gg} r) \cdot X$	ARB5
$r \tilde{\gg} (a(t) \cdot X) = (r \tilde{\gg} a(t)) \cdot X$	ARB6

Table 31. Reduction axioms.

The remaining expressions, $U(a[t])$, $\tilde{U}(a(t))$, $a[t] \gg r$, $a(t) \tilde{\gg} r$ and $r \tilde{\gg} a(t)$ cannot be reduced any further. We can reduce them, however, if we have a fixed (absolute) moment in time. In any state of a system, such a time stamp is provided in the second component. This time stamp can be transferred to the first component by means of the following observation.

8.4 LEMMA.

For every process expression X and time t we have that $\langle X, t\rangle \leftrightarrows \langle t \gg X, t\rangle$.

8.5 ELIMINATION.

Now we can obtain full elimination of auxiliary operators for finite closed expressions of the form $t \gg X$. Due to the previous lemma, this may be considered sufficient. Thus, the remaining cases for the bounded initialization operator are expressions of the form

$$s \gg (a[t] \gg r) \qquad s \gg (a(t) \tilde{\gg} r) \qquad s \gg (r \tilde{\gg} a(t)).$$

For the ultimate delay operator, we are not dealing with process expressions, but with real number expressions. Let : be the operator that represents the time shift on real number expressions. Then, the remaining cases for the ultimate delay operator are expressions of the form

$$s : U(a[t]) \qquad s : \tilde{U}(a(t)).$$

8.6 REAL NUMBER EXPRESSIONS.

Formally, we have a sort RNE of real number expressions, with every element of $\mathbb{R}^{\geq 0}$ as a constant, with binary operators $+, \cdot, -, /$ (here we have to deal with division by 0; we choose the option to put $r/0 = 0$ for all r), and for each atomic action a unary operators $U(a[.])$ and $\tilde{U}(a(.))$, and in addition an operator : from $\mathbb{R}^{\geq 0} \times \mathrm{RNE}$ to RNE. The basic axioms for : are as follows.

$t : r = r$	for $r \in \mathbb{R}^{\geq 0}$
$t : (X \square Y) = (t : X) \square (t : Y)$	for $\square = +, \cdot, /, -$

Table 32. Time shift on real number expressions.

Finally, we have the required elimination axioms (here $s \in \mathbb{R}^{\geq 0}$, $t, r \in \mathrm{RNE}$).

$s : U(a[t]) = s + (s : t)$	ARU3
$s : \tilde{U}(a(t)) = \max\{0, (s : t) - s\}$	ARU4
$s \gg (a[t] \gg r) = (s \gg a[s : t]) \gg (s : r)$	ARB7
$s \gg (a(t) \tilde{\gg} r) = (s \gg a(s : t)) \tilde{\gg} (s : r)$	ARB8
$s \gg (r \tilde{\gg} a(t)) = (s + s : r) \gg a(s : t)$	ARB9
$s \gg (X \parallel Y) = (s \gg X) \parallel (s \gg Y)$	ARB10
$s \gg (X \between Y) = (s \gg X) \between (s \gg Y)$	ARB11
$s \gg (X \mid Y) = (s \gg X) \mid (s \gg Y)$	ARB12
$s \gg \partial_H(X) = \partial_H(s \gg X)$	ARB13

Table 33. Elimination axioms.

8.7 EXAMPLES.

i. $3 \gg (a(7) \parallel b[2]) = (3 \gg a(7)) \parallel (3 \gg b[2]) = a(7) \parallel b(5) = b(5) \cdot a(7)$;

ii. $6 \gg (a(7) \parallel b[2]) = (6 \gg a(7)) \parallel (6 \gg b[2]) = a(7) \parallel b(8) = a(7) \cdot b(8)$.

9. TRANSITION SYSTEMS.

One of the virtues of the availability of axioms for transforming process expressions is that one may view the operational meaning of real time processes in terms of rewriting congruence classes of process expressions. Because we have axioms that explain all operators in terms of + and · one will only need operational rules for these two fundamental operators and the atomic actions. Thus one of the reasons for having the axioms of ACPρ is that these allow to reduce the problem of giving an operational meaning to that for BPAρδ.

9.1 ACTION RELATIONS ON CONGRUENCE CLASSES.

In the case of symbolic process algebra the three rules $a + X \xrightarrow{a} \surd$ and $X \xrightarrow{a} X' \Rightarrow X \cdot Y + Z \xrightarrow{a} X' \cdot Y$ and $X \xrightarrow{a} \surd \Rightarrow X \cdot Y + Z \xrightarrow{a} Y$ are sufficient to define an operational semantics on congruence classes (two rules suffice if we only deal with prefix multiplication instead of general sequential composition). This means that $X \xrightarrow{a} Y$ (resp. $X \xrightarrow{a} \surd$) is operationally valid if and only if $X = a \cdot Y + Z$ (resp. $X = a + Z$) is provable for some Z. Unfortunately in the case of real time more rules are needed because idling, step and termination must be distinguished. It follows that in order to define the operational meaning of BPAρδ one needs one rule for δ, two rules for atomic actions and three rules for sequential composition.

$$t < r \quad \Rightarrow \quad \langle a(r) + x, t\rangle \xrightarrow{a(r)} \langle \surd, r\rangle$$

$$t < s < r \Rightarrow \quad \langle a(r) + x, t\rangle \rightarrow \langle a(r) + x, s\rangle$$

$$t < s < r \Rightarrow \quad \langle \delta(r) + x, t\rangle \rightarrow \langle \delta(r) + x, s\rangle$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle x',r\rangle}{\langle x\cdot y + z,\ t\rangle \xrightarrow{a(r)} \langle x'\cdot y,\ r\rangle}$$

$$\frac{\langle x,t\rangle \rightarrow \langle x,r\rangle}{\langle x\cdot y + z,\ t\rangle \rightarrow \langle x\cdot y + z,\ r\rangle}$$

$$\frac{\langle x,t\rangle \xrightarrow{a(r)} \langle \surd,r\rangle}{\langle x\cdot y + z,\ t\rangle \xrightarrow{a(r)} \langle y,r\rangle}$$

Table 34. Action rules for congruence classes of BPAρδ.

For instance, if we want to derive $\langle a(2) + (b(3)\cdot c(4) + e(5)),\ 0\rangle \xrightarrow{b(3)} c(4)$, we write $a(2) + (b(3)\cdot c(4) + e(5)) = (b(3)\cdot c(4) + e(5)) + a(2) = b(3)\cdot c(4) + (e(5) + a(2))$, and then the expression has the right format for the fifth rule.

For BPAρδ, an operational semantics is as easily presented for the free syntax as for the congruence classes. Indeed a few rules can be removed if one uses congruence classes but there is no fundamental advantage. It is the case however that the axioms of BPA are useful (and in some cases needed) for expressing merge and encapsulation in terms of + and ·.

9.2 REAL TIME TRANSITION SYSTEMS.

We have presented a structured operational semantics definition for closed real time process algebra expressions in an earlier section. Clearly the outcome of that definition is some kind of real time transition system for every process expression to which it is applied. Nevertheless the effort leaves open the question as to what a real time transition might be in general and how the operations of ACPρ have to be interpreted in a general world of real time transition systems. We will consider answers to these questions below. It should be noticed that there are discrepancies between the definitions below and the earlier structured operational semantics. The main difference is that we split a termination step into a regular step plus a termination predicate.

We stress that there are as many definitions of a real time transition system possible as there will be researchers in this area. It is important however to understand the difference between associating a real time transition system to process expressions directly and doing it via an interpretation of the operators in a suitable structure. The point is that the transition systems for which a description by means of a process algebra expression exists constitute a minority only. Now that holds for the symbolic case as well but there is a major difference with the symbolic case: as soon as a general concept of a real time transition system is defined, already for systems that can perform no single atomic action there turn out to be complicated structural possibilities which are all ruled out if one works with interpretations of closed process algebra expressions.

9.3 DEFINITION.

A *real time transition system* (RTTS) over a set of atomic actions A (not including δ) consists of

- a set S of states;
- a root $r \in S$;
- a function $T: S \to \mathbb{R}^{\geq 0}$ that assigns a time to each state;
- a relation $\surd$ on S, that determines all terminated states;
- a relation $idle \subseteq S \times S$ (notation $s \rightarrow s'$);
- a relation $step \subseteq S \times A_T \times S$ (notation $s \xrightarrow{a(t)} s'$).

There are several requirements on an RTTS:

i. If $s \rightarrow s'$ then $T(s) < T(s')$ and for every r with $T(s) < r < T(s')$ there exists a $u \in S$ such that $T(u) = r$ and $s \rightarrow u \rightarrow s'$.

ii. If $s \xrightarrow{a(t)} s'$ then $T(s) < T(s')$, and for every r with $T(s) < r < T(s')$ there exists a $u \in S$ such that $T(u) = r$ and $s \rightarrow u \xrightarrow{a(t)} s'$.

iii. If $s \rightarrow s'$ and $s' \rightarrow s''$ then $s \rightarrow s''$.

iv. If $s \rightarrow s'$ and $s' \xrightarrow{a(t)} s''$ then $s \xrightarrow{a(t)} s''$.

v. $T(r) = 0$.

vi. If $\surd(s)$ then for no action a and state s' we have $s \xrightarrow{a(t)} s'$.

vii. If $\surd(s)$ and $T(s) < r$ then for some s' we have $T(s') = r$ and $s \rightarrow s'$; this means that once a terminal state has been obtained the system can idle forever.

viii. If $\surd(s)$ and $s \rightarrow s'$ then $\surd(s')$.

On an RTTS the signature of ACPρ must be interpreted. We assume that an associative and commutative communication function $|$ on the atomic actions is given which has δ as a zero.

9.4 ATOMIC ACTION.

We start with a description of the transition system belonging to a timed atomic action $a(r)$.

$S = \mathbb{R}^{\geq 0}$ $\quad$ $root = 0$ $\quad$ $T(t) = t$ $\quad$ $\surd(t)$ iff $t \geq r$

$t \rightarrow t'$ iff $t < t' < r$ or $r \leq t < t'$ $\quad$ $t \xrightarrow{a(r)} r$ for $t < r$.

We will always assume that no transitions exist other than the ones mentioned.

Next we consider the timed versions $\delta(r)$ of δ.

$S = \mathbb{R}^{\geq 0}$ $\quad$ $root = 0$ $\quad$ $T(t) = t$ $\quad$ $\surd(t) = false$

$t \rightarrow t'$ iff $t < t' < r$.

9.5 ALTERNATIVE COMPOSITION.

The definition of sum is not quite so straightforward as one might wish. We start with two transition systems $S_1, r_1, T_1, \surd_1, \rightarrow_1, \xrightarrow{a(t)}_1$ and $S_2, r_2, T_2, \surd_2, \rightarrow_2, \xrightarrow{a(t)}_2$ and their sum will be the system $S, r, T, \surd, \rightarrow, \xrightarrow{a(t)}$. Moreover $*$ is supposed to be an object outside both S_1 and S_2.

$S = \{\langle s_1, s_2\rangle \in S_1 \times S_2 \mid T_1(s_1) = T_2(s_2)\} \cup \{\langle s_1, *\rangle \mid s_1 \in S_1\} \cup \{\langle *, s_2\rangle \mid s_2 \in S_2\}$;

$r = \langle r_1, r_2\rangle$;

$T(\langle s_1, s_2\rangle) = T_1(s_1)$, $T(\langle s_1, *\rangle) = T_1(s_1)$, $T(\langle *, s_2\rangle) = T_2(s_2)$;

$\surd(\langle s_1, s_2\rangle)$ iff $(\surd_1(s_1)$ and $\surd_2(s_2))$, $\surd(\langle s_1, *\rangle)$ iff $\surd_1(s_1)$, $\surd(\langle *, s_2\rangle)$ iff $\surd_2(s_2)$;

$\langle s_1, s_2\rangle \rightarrow \langle s_1', s_2'\rangle$ if $s_1 \rightarrow_1 s_1'$ and $s_2 \rightarrow_2 s_2'$,

$\langle s_1, *\rangle \rightarrow \langle s_1', *\rangle$ if $s_1 \rightarrow_1 s_1'$, $\langle *, s_2\rangle \rightarrow \langle *, s_2'\rangle$ if $s_2 \rightarrow_2 s_2'$;

$\langle s_1, s_2\rangle \xrightarrow{a(t)} \langle s_1', *\rangle$ if $s_1 \xrightarrow{a(t)}_1 s_1'$, $\langle s_1, s_2\rangle \xrightarrow{a(t)} \langle *, s_2'\rangle$ if $s_2 \xrightarrow{a(t)}_2 s_2'$,

$\langle s_1, *\rangle \xrightarrow{a(t)} \langle s_1', *\rangle$ if $s_1 \xrightarrow{a(t)}_1 s_1'$, $\langle *, s_2\rangle \xrightarrow{a(t)} \langle *, s_2'\rangle$ if $s_2 \xrightarrow{a(t)}_2 s_2'$.

This definition of sum requires an explanation. The pairs $\langle s_1, s_2\rangle$ denote states in which the system has not yet made a choice. These states can be obtained by waiting from the root. The first atomic action (if any) imposes a choice for the transition system from which it was taken.

9.6 PARALLEL COMPOSITION.

The definition of merge is straightforward indeed. Again we start with two transition systems $S_1, r_1, T_1, \surd_1, \rightarrow_1, \xrightarrow{a(t)}_1$ and $S_2, r_2, T_2, \surd_2, \rightarrow_2, \xrightarrow{a(t)}_2$ and their merge will be the system $S, r, T, \surd, \rightarrow, \xrightarrow{a(t)}$.

$S = \{\langle s_1, s_2\rangle \in S_1 \times S_2 \mid T_1(s_1) = T_2(s_2)\}$;

$r = \langle r_1, r_2\rangle$;

$T(\langle s_1, s_2\rangle) = T_1(s_1)$;

$\surd(\langle s_1, s_2\rangle)$ iff $\surd_1(s_1)$ and $\surd_2(s_2)$;

$\langle s_1, s_2\rangle \rightarrow \langle s_1', s_2'\rangle$ if $s_1 \rightarrow_1 s_1'$ and $s_2 \rightarrow_2 s_2'$;

$\langle s_1, s_2\rangle \xrightarrow{a(t)} \langle s_1', s_2'\rangle$ if: (1) $s_1 \xrightarrow{a(t)}_1 s_1'$ and $s_2 \rightarrow_2 s_2'$ or (2) $s_2 \xrightarrow{a(t)}_2 s_2'$ and $s_1 \rightarrow_1 s_1'$ or (3) $s_1 \xrightarrow{b(t)}_1 s_1'$ and $s_2 \xrightarrow{c(t)}_2 s_2'$ and $b \mid c = a$ for some atomic actions b and c.

9.7 SEQUENTIAL COMPOSITION.

Next comes sequential composition. Again we start with two transition systems S_1, r_1, T_1, $\surd_1$, $\rightarrow_1$, $\xrightarrow{a(t)}_1$ and S_2, r_2, T_2, $\surd_2$, $\rightarrow_2$, $\xrightarrow{a(t)}_2$ and their product will be the system S, r, T, $\surd$, $\rightarrow$, $\xrightarrow{a(t)}$.

$S = \{\langle s_1, * \rangle \mid s_1 \in S_1\} \cup \{\langle *, s_2 \rangle \mid s_2 \in S_2\}$;

$r = \langle r_1, * \rangle$;

$T(\langle s_1, * \rangle) = T_1(s_1)$, $T(\langle *, s_2 \rangle) = T_2(s_2)$;

$\surd(\langle s_1, * \rangle) =$ false, $\surd(\langle *, s_2 \rangle)$ iff $\surd_2(s_2)$;

$\langle s_1, * \rangle \rightarrow \langle s_1', * \rangle$ if $s_1 \rightarrow_1 s_1'$ and not $\surd_1(s_1)$;

further $\langle s_1, * \rangle \rightarrow \langle *, s_2 \rangle$ if there exists s_1' such that

$T_1(s_1') \leq T_2(s_2)$, $\surd_1(s_1')$, $r_2 \rightarrow_2 s_2$, and either $s_1 \equiv s_1'$ or $s_1 \rightarrow_1 s_1'$;

and finally $\langle *, s_2 \rangle \rightarrow \langle *, s_2' \rangle$ if $s_2 \rightarrow_2 s_2'$;

$\langle s_1, * \rangle \xrightarrow{a(t)} \langle s_1', * \rangle$ if $s_1 \xrightarrow{a(t)}_1 s_1'$;

further $\langle s_1, * \rangle \xrightarrow{a(t)} \langle *, s_2 \rangle$ if there exists s_1' such that

$T_1(s_1') \leq T_2(s_2)$, $\surd_1(s_1')$, $r_2 \xrightarrow{a(t)}_2 s_2$, and either $s_1 \equiv s_1'$ or $s_1 \rightarrow_1 s_1'$;

and finally $\langle *, s_2 \rangle \xrightarrow{a(t)} \langle *, s_2' \rangle$ if $s_2 \xrightarrow{a(t)}_2 s_2'$.

9.8 OTHER OPERATORS.

The definition of encapsulation ∂_H on S_1, r_1, ... is easy indeed:

$S = S_1$; $r = r_1$; $T(s) = T_1(s)$; $\surd(s)$ iff $\surd_1(s)$; $s \rightarrow s'$ iff $s \rightarrow_1 s'$;

$s \xrightarrow{a(t)} s'$ iff $a \notin H$ and $s \xrightarrow{a(t)}_1 s'$.

Then we are left with the auxiliary operators ultimate delay, bounded initialization, absolute time shift, left merge and communication merge. All of these can be given similar operational definitions in transition systems.

9.9 BISIMULATION AS A CONGRUENCE.

Of course an appropriate development of the theory requires that we define bisimulation equivalence on these transition systems and establish that all operations respect bisimulation equivalence whence bisimulation becomes a congruence. Unfortunately, carrying out such proofs in acceptable detail is far from trivial. We consider that in itself to be an open area of research. For the time being however, the introduction of transition systems and operators on them merely serves the purpose of strengthening the mechanical intuitions about real time processes in the ACP framework.

The contribution of this work lies in developing a syntax for real time processes that is in line with the syntax of ACP and in the proposal of a workable set of axioms for this syntax. A mathematical analysis of the model theory of this syntax is a matter for future work. It should be stressed however that such foundational work is only justified if one is convinced of the usefulness of the syntax. Only if the formalism ACPρ is considered a promising language for the analysis of real time computing mechanisms the tremendous effort of a meticulous analysis of its model theory becomes a worthwhile investment.

9.10 ABSTRACTION OF INTERNAL STEPS.

There is an obvious way to define abstraction from internal steps in a transition system. Indeed let τ_I denote an abstraction operator that abstracts from internal steps. Then in order to apply it on a real time transition system all that has to be done is to replace transitions $s \xrightarrow{a(t)} s'$ with $a \in I$ by $s \rightarrow s'$, and apply transitive closure on the wait and step transitions. This gives rise to an equivalence relation on terms that turns out not be a congruence relation (this also happens in the untimed case with weak bisimulation): we obtain the same transition system for $\tau(1)\cdot a(2)$ and $a(2)$, but different transition systems for $\tau(1)\cdot a(2) + b(2)$ and $a(2) + b(2)$ (in the former term, the choice for a or b is made at time 1, in the latter term this choice is not made until time 2). We can fix this problem, if to all states in a transition systemwe add the information whether or not the process has started at that point.

Furthermore, we can obtain a complete axiomatization of internal actions by means of the central law:

$$s < t \;\&\; U(x) > t \;\&\; U(y) \leq t \;\Rightarrow\; a(s)\cdot(\tau(t)\cdot x + y) = a(s)\cdot(t \gg x + y).$$

All these matters form the subject of ongoing research.

9.11 PICTORIAL REPRESENTATIONS.

For some very simple real time transition systems it is possible to provide graphical representations in which one may read off the transitions from the picture. Using these pictures one immediately finds examples of transition systems for which there is no corresponding process expression (unless one introduces an abstraction mechanism).

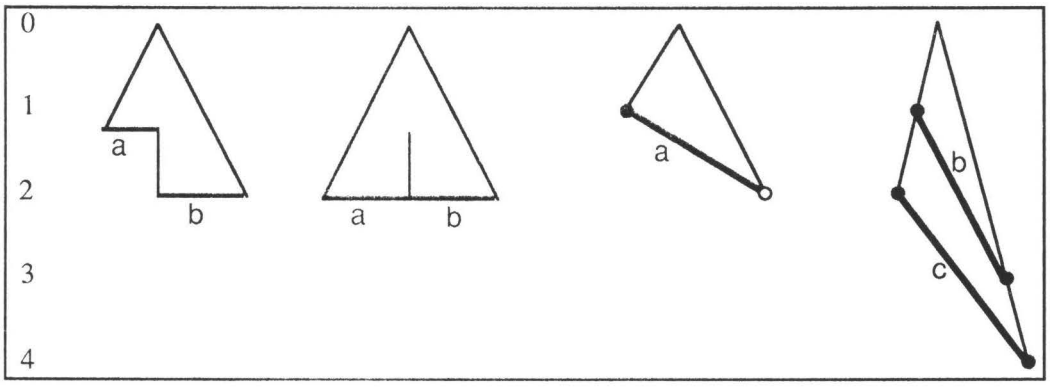

FIGURE 2.

In the drawings in fig. 2, the root is the top point. The vertical axis is the time axis (so transitions must always end lower than they start). Transitions may never cross a line. Transitions that end in a thick line are steps. Bottom points on a thick line are $\surd$ points. We see that the first picture can be represented by term $a(1) + b(2)$, that the second has no representation without using internal steps (but can be represented by $\tau(1)\cdot a(2) + \tau(1)\cdot b(2))$, the third can be represented by $\backslash i(_{t\in[1\backslash,2\backslash)},, a(t))$ and the fourth by $(\int_{t\in[1,3]} b(t)) \cdot (\int_{t\in[2,4]} c(t))$.

9.12 TIME REMOVAL.

Suppose we design a system in symbolic process algebra notation first. Suppose the design has the following form: $\partial_H(P_1 \parallel P_2 \parallel P_3)$. Here we assume that the components P_i have been specified as regular processes, using linear process specifications over BPA. As a next step we may provide time stamps to the actions of the components P_1, P_2, P_3. This can be done as in the example of the alternating bit protocol (see the following section), obtaining Q_1, Q_2, Q_3. Already at this point one may ask whether there is any mathematical connection between the respective P_i and Q_i. We propose a way to transform a real time transition system to a symbolic transition system. This transformation called TR (time removal) is such that in many cases one may expect that $TR(RTS(\langle Q_i, 0\rangle))$ is bisimilar to P_i. Thus, adding timing information to a symbolic design component P_i should be done in such a way that performing time abstraction on the resulting real time transition system leads back to the symbolic design component (modulo bisimulation at least). So one may say that a real time implementation of P_i by Q_i is correct if $TR(RTS(\langle Q_i, 0\rangle)) = P_i$ (modulo bisimulation).

Next suppose that in symbolic process algebra one proves that

$$P = \tau_I \circ \partial_H(P_1 \parallel P_2 \parallel P_3).$$

Then, using time removal it is possible to express that the implementation of the P_i by the Q_i is sound with respect to the symbolic verification $P = \tau_I \circ \partial_H(P_1 \parallel P_2 \parallel P_3)$ as follows:

Let X be the real time transition system corresponding to $\tau_I \circ \partial_H(Q_1 \parallel Q_2 \parallel Q_3)$. Then it is required that TR(X) is bisimilar to P.

Notice that there is no implicit guarantee whatsoever that the real time implementation of a symbolic system specification obtained from correct real time implementations of its components is correct with respect to some given symbolic verification of the design. In fact, in many cases it will be incorrect. The development of general conditions that guarantee this form of soundness is perhaps a relevant topic for further work (though we see as yet no indication that nice and convincing conditions can be found).

9.13 DEFINITION OF TIME REMOVAL ON REAL TIME TRANSITION SYSTEMS.

The mapping of a real time transition system to a symbolic transition system proceeds through various stages. Let a system S, r, T, $\surd$, $\rightarrow$, $\overset{a(t)}{\rightarrow}$ be given. We proceed in three stages.

STAGE 1. We introduce two new predicates on S: ACT (active) and INACT (inactive). Together with $\surd$ these predicates cover all states. Moreover the three predicates are mutually exclusive.

ACT contains all states in the transition system outside $\surd$, where the process has started and from which a transition labeled with an action is possible, or from which a transition leading to a terminated state is possible, i.e. all s such that not $\surd(s)$, such that a step has occurred before s (possibly internal) and either for some s' and a(t) there is a transition $s \overset{a(t)}{\rightarrow} s'$ or for some $s' \in \surd$ we have $s \rightarrow s'$.

$INACT = S - (ACT \cup \surd)$.

Notice that the collection INACT contains states in which the system has essentially deadlocked (there are no waiting steps possible any more, or stated more generally: there is a finite upper bound to the possible waiting time from that state), as well as states where the system is in a livelock (indefinite waiting is possible and even unavoidable) and in addition to this, states that are somehow in between

(there is no upper bound to possible waiting but the system may develop into a deadlocked situation). Using these conventions we stay close to what happens in the bisimulation model of ACP_τ (see [BBK87]) where livelock and deadlock are not distinguished; this simply means that all inactive processes are mapped to δ.

STAGE 2. In the second step the actions and states of the system are modified. We add two states δ and $\surd$ to S (assuming that these objects were not in S and renaming some states in S otherwise; this is allowed because we work modulo bisimulation equivalence). Further, we use the silent action τ as a label for transitions. Notice that $\tau \notin A_\delta$ so this cannot lead to confusion.

STAGE 3. Now all transitions are modified, obtaining the symbolic transition system TS* from a given RTTS TS. For each transition α of TS it is decided whether or not to put a corresponding transition α^* in TS*.

- if α is $s \to s'$ then:
  CASE 1: if $\surd(s)$ or INACT(s) there is no α^*;
  CASE 2: if ACT(s) then: CASE 2.1: if $\surd(s')$ then $\alpha^* = s \xrightarrow{\tau} \surd$;
  CASE 2.2: if INACT(s') then $\alpha^* = s \xrightarrow{\tau} \delta$;
  CASE 2.3: if ACT(s') then $\alpha^* = s \xrightarrow{\tau} s'$;
- if α is $s \xrightarrow{a(t)} s'$ then: CASE 1: if $\surd(s')$ then α^* is $s \xrightarrow{a} \surd$;
  CASE 2: if INACT(s') then α^* is $s \xrightarrow{a} \delta$;
  CASE 3: if ACT(s') then α^* is $s \xrightarrow{a} s'$.

We denote this mapping from real time transition systems to symbolic transition systems by TR (time removal). This is a complicated operation for which many alternative definitions are conceivable. We do not suggest that this operation is canonical in any way. It is not clear what intrinsic requirements one may wish to impose on such time removing transformations. Nevertheless, with some examples it becomes clear what technical role TR may play.

For a closed symbolic process expression P we call the transition system assigned to P STS(P). In fact this is rather vague, due to the many ways in which an operational semantics can be given, but working modulo bisimulation, and with the understanding that terminating transitions are denoted by arrows ending in $\surd$, and that every transition has a label in A, the concept of STS(P) is reasonably stable.

9.14 SYMBOLIC PROCESS EQUIVALENCE.
Let $P = a(2) \cdot b(3)$, then TR(RTS(⟨P, 0⟩)) bisimulates with $STS(a \cdot b)$. Similarly, if $Q = a(3) \cdot b(5)$, then TR(RTS(⟨Q, 0⟩)) bisimulates with $STS(a \cdot b)$.

In that way we find an equivalence relation, *symbolic process equivalence* on real time processes, as follows: real time process expressions P and Q are equivalent in this sense if TR(RTS(⟨P, 0⟩)) rooted τ-bisimulates with TR(RTS(⟨Q, 0⟩)) (the congruence relation rτ-bisimulation on symbolic process expression is defined e.g. in [BBK87]; it is the same relation as weak observational congruence of [M80]). The symbolic process equivalence relation is nowhere near being a congruence relation (there are problems with just about every process algebra operator except encapsulation).

9.15 EXAMPLES.

i. $TR(RTS(\langle b(2) + c(3), 0\rangle))$ $r\tau$-bisimulates with $STS(b + c)$.

ii. $TR(RTS(\langle a(1)\cdot(b(2) + c(3)), 0\rangle))$ $r\tau$-bisimulates with $STS(a\cdot(b + \tau\cdot c))$.

iii. Let $P_0(n) = a(n)\cdot P_0(n+1)$ and $Q_0 = a\cdot Q_0$ then $TR(RTS(\langle P_0(1), 0\rangle))$ $r\tau$-bisimulates with $STS(Q_0)$.

iv. Let $P_1(n) = (b(n)\cdot c(n+1) + a(n+2))\cdot P_1(n+2)$ and $Q_1 = b\cdot c\cdot Q_1 + \tau\cdot a\cdot Q_0$ (Q_0 from iii), then $TR(RTS(\langle P_1(1), 0\rangle))$ $r\tau$-bisimulates with $STS(b\cdot c\cdot Q_1 + a\cdot Q_0)$.

v. Let $P_2(n) = a(n) + \tau(n)\cdot P_2(n+1)$, then $TR(RTS(\langle P_2(1), 0\rangle))$ $r\tau$-bisimulates with $STS(\tau\cdot a)$.

vi. Let $P_3(n) = \tau(n) \cdot P_3(n + 1)$, then $TR(RTS(\langle P_3(1), 0\rangle))$ $r\tau$-bisimulates with $STS(\tau\cdot\delta)$.

10. PROCESS SPECIFICATIONS.

In this final section, we give some further examples of process specifications, and make a few concluding remarks.

10.1 EXAMPLE.

A FIFO queue with unbounded capacity and fixed transition time for each message.

$$Q = \int_{v\in\mathbb{R}^{\geq 0}} \sum_{d\in D} r1(d)[v] \cdot (s2(d)[\Delta] \parallel Q)$$

10.2 EXAMPLE.

A FIFO queue that can input messages at any rate and outputs them at a rate of one per time unit (as long as the queue is not empty). $Q_\sigma(v)$ denotes this Q in a state in which it contains the data elements in the list σ and v denotes the most recent time at which it has either presented its previous output or at which it has become non-empty by reading a message in empty state. In the list σ the data are coded in such a way that that last message is positioned at the leftmost position in the sequence.

$$Q_\varepsilon(v) = \int_{u=(v,\omega)} \sum_{d\in D} r1(d)(u) \cdot Q_d(u)$$

$$Q_{\sigma^*a}(v) = \int_{u=(v,v+1)} \sum_{d\in D} r1(d)(u) \cdot Q_{d^*\sigma^*a}(v) + s2(a)(v+1) \cdot Q_\sigma(v+1) + \sum_{d\in D} (s2(a)(v+1)\&r1(d)(v+1)) \cdot Q_{d^*\sigma}(v+1).$$

Notice that this process is defined by means of an infinite number of equations, one for each Q_σ. We have not found a way to restrict the number of equations to finitely many.

10.3 ALTERNATING BIT PROTOCOL.

We first provide a specification of an alternating bit protocol as given in [BK86]. Then we will add real time information about each atomic action involved.

In this protocol e denotes an error value, $e \notin D$. db denotes the pair of a datum d from D and a boolean value b (the alternating bit). With i we denote an internal action that is used to encode a non-

deterministic choice between correct and incorrect transition by the channels. In ACP the specification is as follows:

Protocol

$ABP = \partial_{H(3,4,5,6)}(A \parallel K \parallel L \parallel B)$

where

sender

$A = RM_0$

$RM_b = \sum_{d \in D} r1(d) \cdot SF_{db}$

$SF_{db} = s3(db) \cdot RA_{db}$

$RA_{db} = [r5(1\text{-}b) + r5(e)] \cdot SF_{db} + r5(b) \cdot RM_{1\text{-}b}$

channels

$K = \sum_{f \in D} r3(f) \cdot K_f$

$K_f = [i \cdot s4(e) + i \cdot s4(f)] \cdot K$

$L = \sum_{b \in BOOL} r6(b) \cdot L_b$

$L_b = [i \cdot s5(e) + i \cdot s5(b)] \cdot L$

receiver

$B = RF_0$

$RF_b = [\sum_{d \in D} r4(d(1\text{-}b)) + r4(e)] \cdot SA_{1\text{-}b} + \sum_{d \in D} r4(db) \cdot SM_{db}$

$SA_b = s6(b) \cdot RF_{1\text{-}b}$

$SM_{db} = s2(d) \cdot SA_b$

In ACP one may prove that ABP is in fact a sequential system in the sense that in no state, message passing is enabled along more than one port. Now we will provide a timed version of the protocol. The delays that are introduced are taken in an arbitrary way, for the sake of the example.

Again this system will turn out to be a sequential one in the sense that no two ports can be used at the same time.

protocol

$ABP = \partial_{H(3,4,5,6)}(A \parallel K \parallel L \parallel B)$

where

sender

$A = RM_0$

$RM_b = \int_{v \in \mathbb{R}^{\geq 0}} \sum_{d \in D} r1(d)[v] \cdot SF_{db}$

$$SF_{db} = s3(db)[0.001] \cdot RA_{db}$$

$$RA_{db} = \int_{v \in \mathbb{R}^{\geq 0}} [r5(1\text{-}b)[v] + r5(e)[v]] \cdot SF_{db} + r5(b)[v] \cdot RM_{1\text{-}b}$$

channels

$$K = \int_{v \in \mathbb{R}^{\geq 0}} \sum_{f \in D} r3(f)[v] \cdot K_f$$

$$K_f = \int_{w \in [0.009, 0.011]} (i[w] \cdot s4(e)[0.0001] + i[w] \cdot s4(f)[0.0001]) \cdot K$$

$$L = \int_{v \in \mathbb{R}^{\geq 0}} \sum_{b \in BOOL} r6(b)[v] \cdot L_b$$

$$L_b = \int_{w \in [0.009, 0.011]} (i[w] \cdot s5(e)[0.0001] + i[w] \cdot s5(b)[0.0001]) \cdot L$$

receiver

$$B = RF_0$$

$$RF_b = \int_{v \in \mathbb{R}^{\geq 0}} \sum_{d \in D} (r4(d(1\text{-}b))[v] + r4(e)[v]) \cdot SA_{1\text{-}b} + \sum_{d \in D} r4(db)[v] \cdot SM_{db}$$

$$SA_b = s6(b)[0.008] \cdot RF_{1\text{-}b}$$

$$SM_{db} = s2(d)[0.01] \cdot s6(b)[0.012] \cdot RF_{1\text{-}b}.$$

Notice that the alternating bit protocol is a regular processes. For any reasonable definition of regularity in real time systems, also the real time specification of the alternating bit protocol will be a regular process.

10.4 Timed vs. untimed specifications.

In the example of the alternating bit protocol we illustrate an obvious technique that can be used to turn a symbolic specification of a system into a real time specification.

i. All read actions can happen at any time;

ii. The time at which a read action happens is remembered in the state (i.e.: it determines the timing of the next action);

iii. Time constraints are allowed on send actions only. These may happen at fixed moments in time computable from the data stored in the state (including the timing of the previous action) or in intervals for which the bounds can be computed from the mentioned data.

When taking these rules into account one obtains real time process specifications in which there is a complete separation between time and data. For such specifications it is possible to remove all timing information, thus obtaining a symbolic process specification as indicated in section 9.

10.5 CONCLUDING REMARK.

This report presents an introduction to the notations and equations for a real time version of process algebra (in the style of ACP). Many more features can be incorporated without substantial difficulty. We mention: state operators, asynchronous communication, put & get (unreliable communication), mode transfer, realization of successful handshaking communication if sender or receiver is timesharing a different task (leading to interrupt handling without use of a priority mechanism), process creation, process forking, a constant for chaos.

Matters that seem to be hard to generalize are for instance the notion of a finitely branching process, priorities, regular (finite state) processes and the notion of a linear system of equations. Real time system verification in the setting of ACPρ is an open area altogether though some significant notions of correctness have been identified in this paper.

ACKNOWLEDGEMENTS.

The authors gratefully acknowledge many suggestions for improvements that were done by the anonymous referees, Steven Klusener (CWI Amsterdam), Hans Mulder (University of Nijmegen) and Kim Regan (University of Sussex).

REFERENCES.

[BB90] J.C.M. BAETEN & J.A. BERGSTRA, *Real space process algebra,* report P9005, Programming Research Group, University of Amsterdam 1990.

[BBK87] J.C.M. BAETEN, J.A. BERGSTRA & J.W. KLOP, *On the consistency of Koomen's Fair Abstraction Rule,* Theor. Comp. Sci. 51, 1987, pp. 129-176.

[BK84] J.A. BERGSTRA & J.W. KLOP, *Process algebra for synchronous communication,* Inf. & Control 60, 1984, pp. 109-137.

[BK86] J.A. BERGSTRA & J.W. KLOP, *Process algebra: specification and verification in bisimulation semantics,* in: Math. & Comp. Sci. II (M. Hazewinkel, J.K. Lenstra & L.G.L.T. Meertens, eds.), CWI Monographs 4, North-Holland, Amsterdam 1986, pp. 61-94.

[DS89] J. DAVIES & S. SCHNEIDER, *An introduction to timed CSP,* report PRG-75, Oxford University Computing Laboratory, Oxford UK 1989.

[GB87] R. GERTH & A. BOUCHER, *A timed failures model for extended communicating processes,* in: Proc. 14th ICALP, Karlsruhe (Th. Ottmann, ed.), Springer LNCS 267, 1987, pp. 95-114.

[GV87] R.J. VAN GLABBEEK & F.W. VAANDRAGER, *Petri net models for algebraic theories of concurrency (extended abstract),* Proc. PARLE Vol. II (J.W. de Bakker, A.J. Nijman & P.C. Treleaven, eds.), Springer LNCS 259, 1987, pp. 224-242.

[G89] J.F. GROOTE, *Transition system specifications with negative premises,* report CS-R8950, CWI, Amsterdam 1989; *Extended abstract* in: Proc. CONCUR 90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), Springer LNCS 458, 1990, pp. 332-341.

[G90] J.F. GROOTE, *Specification and verification of real time systems in ACP,* report CS-R9015, CWI, Amsterdam 1990; to appear in Proc. 10th Int'l IFIP Symp. on Protocol Specification, Testing and Verification, Ottawa 1990.

[JM86] F. JAHANIAN & A. MOK, *Safety analysis of timing properties in real-time systems,* IEEE Transactions on Software Engineering 12, 1986, pp. 890-904.

[KSRGA88] R. KOYMANS, R.K. SHYAMASUNDAR, W.P. DE ROEVER, R. GERTH & S. ARUN-KUMAR, *Compositional semantics for real-time distributed computing,* Inf. & Comp. 79, 1988, pp. 210-256.

[M80] R. MILNER, *A calculus of communicating systems,* Springer LNCS 92, 1980.

[MT90] F. MOLLER & C. TOFTS, *A temporal calculus of communicating systems,* in: Proc. CONCUR 90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), Springer LNCS 458, 1990, pp. 401-415.

[NRSV90] X. NICOLLIN, J.-L. RICHIER, J. SIFAKIS & J. VOIRON, *ATP: an algebra for timed processes,* report RT-C16, Projet Spectre, IMAG, 1990, to appear in: Proc. IFIP Conf. Progr. Concepts and Methods, Sea of Gallilee 1990.

[R89] G.M. REED, *A hierarchy of domains for real-time distributed computing,* to appear in Proc. MFPLS '89.

[RR86] G.M. REED & A.W. ROSCOE, *A timed model for communicating sequential processes,* Theor. Comp. Sci. 58, 1988, pp. 249-261.

[RR87] G.M. REED & A.W. ROSCOE, *Metric spaces as models for real-time concurrency,* in: Proc. MFPLS '87, Springer LNCS 298, 1988.

[T81] A.S. TANENBAUM, *Computer networks,* Prentice Hall International, 1981.

[V90] F.W. VAANDRAGER, *Algebraic techniques for concurrency and their application,* Ph.D. Thesis, University of Amsterdam 1990.

[ZL87] A. ZWARICO & I. LEE, *A syntax and semantics for deterministic real-time computing,* technical report, Dept. of Comp. & Inf. Sci., University of Pennsylvania, Philadelphia, 1987.