Asynchronous Communication in Real Space Process Algebra

J.C.M. Baeten*
Department of Software Technology, CWI,
P.O.Box 4079, 1009 AB Amsterdam, The Netherlands
and
Programming Research Group, University of Amsterdam,
P.O.Box 41882, 1009 DB Amsterdam, The Netherlands

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

A version of classical real space process algebra is given in which messages travel with constant speed through a three-dimensional medium. It follows that communication is asynchronous and has a broadcasting character. A state operator is used to describe asynchronous message transfer and a priority mechanism allows to express the broadcasting mechanism. As an application, a protocol is specified in which the receiver moves with respect to the sender.

Key words & Phrases: process algebra, real time process algebra, real space process algebra, asynchronous communication, broadcasting, state operator, priorities.
Note: Partial support received by ESPRIT basic research action 3006, CONCUR, and by RACE contract 1046, SPECS. This document does not necessarily reflect the views of the SPECS consortium.

1. INTRODUCTION.

Our aim is to extend real time process algebra (see [BAB91a]) to classical (i.e. non-relativistic) real space process algebra (see [BAB91b]) in such a way that motion of processes can be taken into account. Although a rigorous proof is absent, it seems impossible to express communication between processes moving in three-dimensional space with the primitives of [BAB91a] and [BAB91b]. The difficulty arises because both papers use synchronous communication whereas motion of processes seems to call for asynchronous communication: if process P at place/time (x,t) sends d to Q, then Q may receive d at place/time (y,r) provided $|x - y| = v \cdot (r - t)$. Here $|x - y|$ is the distance between x and y and v is the message transmission velocity in the medium connecting P and Q. If P and Q are at rest we find $r = |x - y|/v + t$ and this formula can be used in process expressions in the style of [BAB91a] and [BAB91b].

However, if y depends on time, the equation $|x(t) - y(r)| = v \cdot (r - t)$ has to be solved in order to determine the time r at which message d will be received. In general, no closed form can be found for r. Even worse, the motion of Q (and so y) may depend on actions of the system after t (the time at which d is sent).

---

* Author's current affiliation: Department of Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.

In order to study communication between moving processes we assume that messages travel with speed v through space in all directions. I.e. after d is sent at (x,t) it travels through space as an expanding spherical wave. For simplicity we assume that the message can be detected at any distance. It is probably not difficult to describe a decrease in loudness of the signal together with a threshold mechanism for receivers.

Now according to [BEKT85] asynchronous communication in process algebra can be modeled using auxiliary operators. Subsequently, these operators have been studied in detail in [BAB88] where they are called state operators. In [BAB91c], it is indicated how any state operator definition of untimed process algebra can be extended to real time process algebra over finitely many locations, provided a partial ordering on locations is given. Here, we will consider special state operators (as in [BEKT85]) for which the action and effect functions depend on place and time, but that satisfy a commutativity requirement for actions that happen at the same instant of time.

The state operators λ_V^c (state operator λ of asynchronous channel c, e.g. a radio frequency, in state V) are parametrised by finite collections V of triples $\langle d, x, t\rangle$, d a message in D, x a point in space ($x \in \mathbb{R}^3$), t a time (a non-negative real, $t \in \mathbb{R}^{\geq 0}$). If $\langle d, x, t\rangle \in V$, this indicates that $\lambda_V^c(X)$ provides an environment for X in which along channel c a datum d was sent at place x and time t. The asynchronous send actions $c\uparrow d(x,t)$ will have the effect that $\langle d, x, t\rangle$ is added to V. In the scope of λ_V^c a process X can perform the action $c\downarrow d(y,r)$ (asynchronous read along channel c of datum d at place y and time r) provided $|y - x| = v\cdot(r - t)$.

Because the message d, after being sent by $c\uparrow d(x,t)$, travels in a spherical wave, it can be received more than once and hence the communication mechanism is of a broadcasting nature. As pointed out in [BE85] (see also [BAW90]), a broadcasting mechanism in process algebra calls for the use of the priority operator of [BABK86]. In the real time and space case this priority operator will express maximal progress with respect to some actions as well.

We can summarise this discussion as follows: P and Q, traveling through space, can communicate by performing actions $c\uparrow d(x,t)$ and $c\downarrow d(y,r)$. This works in a context $\lambda_V^c(P \parallel Q)$. Unsuccessful asynchronous reads are blocked by the action function of the state operator. In order to ensure successful reception of the messages a priority operator $\theta_{c\Downarrow D}$ is needed (here $c\Downarrow D$ contains all effectuated $c\downarrow d$ actions). This operator gives priority to all asynchronous communications at port c. We will allow synchronous communications as well. Unsuccessful synchronous reads and sends at channels in H are blocked by the encapsulation operator ∂_{srHD} ($srHD$ contains all synchronous send and receive actions at ports in H). Thus we are led to process expressions of the form:

$$\partial_{srHD} \circ \theta_{c\Downarrow D} \circ \lambda_V^c(P \parallel Q).$$

For instance, a concurrent alternating bit protocol with moving sender and receiver will take the following form:

$$\partial_{srHD}\big(\theta_{1\Downarrow D} \circ \lambda_V^1(P_1 \parallel Q_1) \parallel \theta_{2\Downarrow D} \circ \lambda_V^2(P_2 \parallel Q_2)\big).$$

After studying the asynchronous communication primitives we provide a brief discussion on how to model asynchronous message transfer on the basis of the synchronous communication primitives of $ACP\rho\sigma$. We find that process creation [BE90] and mode transfer [BE89] are helpful in the description of certain asynchronous media. Real time and space versions of these process constructors are provided and various examples are given.

So, besides giving an account of asynchronous communication in a real space setting, this paper also gives the real time equations for several additional features of ACP: the priority operator, state operators with uncountable state space, process creation and both mode transfer operators. Except for the mode transfer operators, these features are all covered in chapter 6 of [BAW90]. The mode transfer operators were not included there, because their equations are not fully satisfactory in the untimed case.

The real time setting provides a clearer picture and the equations given below seem perfectly adequate to us.

By now, there is much work on process algebras that encorporate notions of time (see e.g. [RR88], [MT90], [J91a]). However, there is not much work that also involves real space or the use of locations. Besides papers already mentioned, we only know of JEFFREY [J91b], MURPHY [MU91].

Finally, we remark that we only consider *concrete* process algebra here: there is no concept of a silent or empty step.

ACKNOWLEDGEMENT. We thank Willem Jan Fokkink (CWI Amsterdam) for pointing out that an earlier version of equation TH3 in 4.1 was incorrect. We thank Alan Jeffrey (Chalmers University) for stimulating discussions on maximal progress and priority locks.

2. REAL SPACE PROCESS ALGEBRA.

We give a brief review of classical real space process algebra as introduced in [BAB91b], but we use the timed deadlock of [BAB91a,BAB91c] instead of the untimed deadlock of [BAB91b]. The difference with [BAB91b] is that there, we had a finite set of locations, and here, every element of three-dimensional space is a location. We give an operational semantics in the style of KLUSENER [K91].

2.1 ATOMIC ACTIONS.

We start from a set A of (symbolic) atomic actions. The set of atomic actions with space and time, AST is now generated by

$$\{a(x,t) \mid a \in A, x \in \mathbb{R}^3, t \in \mathbb{R}^{\geq 0}\} \cup \{\delta(t) \mid t \in \mathbb{R}^{\geq 0}\}.$$

It will be useful to consider also the set of atomic actions with only space parameter, i.e. the set AS generated by

$$\{a(x) \mid a \in A, x \in \mathbb{R}^3\} \cup \{\delta\}.$$

We use $a(x)(t)$ as an alternative notation for $a(x,t)$.

2.2 MULTI-ACTIONS.

Multi-actions are process terms generated by actions with space parameter and the *synchronisation function* &. Multi-actions contain actions that occur synchronously at different locations. For α,β,γ elements of AS, we have the following conditions on the synchronisation function (table 1). Further, $x \in \mathbb{R}^3$, $a,b \in A$.

$\alpha \& \beta = \beta \& \alpha$	LO1
$\alpha \& (\beta \& \gamma) = (\alpha \& \beta) \& \gamma$	LO2
$\delta \& \alpha = \delta$	LO3
$a(x) \& b(x) = \delta$	LO4

Table 1. Synchronisation function on AS.

Using the axioms of table 1, each multi-action can be reduced to one of the following two forms:

- δ,
- $a_1(x_1) \& \ldots \& a_n(x_n)$, with all points x_i different, all $a_i \in A$.

Next, we have the *communication function* $\mid$. In table 2, $a,b,c \in A$, $\alpha,\beta,\gamma \in AS$. In order to state axiom CL7, we need an auxiliary function locs, that determines the set of points (locations) of a multi-action.

$a \mid b = b \mid a$	C1
$a \mid (b \mid c) = (a \mid b) \mid c$	C2
$\delta \mid a = \delta$	C3
$\alpha \mid \beta = \beta \mid \alpha$	CL1
$\alpha \mid (\beta \mid \gamma) = (\alpha \mid \beta) \mid \gamma$	CL2
$\delta \mid \alpha = \delta$	CL3
$a(x) \mid b(x) = (a \mid b)(x)$	CL4
$a(x) \mid (b(x) \,\&\, \beta) = (a \mid b)(x) \,\&\, \beta$	CL5
$(a(x) \,\&\, \alpha) \mid (b(x) \,\&\, \beta) = (a \mid b)(x) \,\&\, (\alpha \mid \beta)$	CL6
$locs(\alpha) \cap locs(\beta) = \varnothing \Rightarrow \alpha \mid \beta = \alpha \,\&\, \beta$	CL7
$locs(\delta) = \varnothing$	LOC1
$locs(a(x)) = \{x\}$	LOC2
$x \notin locs(\alpha), locs(\alpha) \neq \varnothing \Rightarrow$ $locs(a(x) \,\&\, \alpha) = locs(\alpha) \cup \{x\}$	LOC3

Table 2. Communication function on AS.

2.3 TIMED MULTI-ACTIONS.

It is now straightforward to extend the definition of the synchronisation and communication functions to timed multi-actions. In table 3, $\alpha,\beta \in AS$.

$t \neq s \Rightarrow \alpha(t) \mid \beta(s) = \delta(\min(t,s))$	CL8
$\alpha(t) \mid \beta(t) = (\alpha \mid \beta)(t)$	CL9

Table 3. Communication function on AST.

2.4 BASIC PROCESS ALGEBRA.

Process algebra (see [BEK84, BAW90]) starts from a given *action alphabet*, here AST. Elements of AST are constants of the sort P of *processes*. The theory Timed Basic Process Algebra with Deadlock (BPA$\rho\delta$) has two binary operators $+,\cdot: P \times P \to P$; $+$ stands for alternative composition and $\cdot$ for sequential composition. Moreover, there is the additional operator $\gg: \mathbb{R}^{\geq 0} \times P \to P$, 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.

$X + Y = Y + X$	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 4. BPA.

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

Table 5. Additional axioms of BPA$\rho\delta$.

BPA$\rho\delta$ has the axioms from table 4 and 5 ($\alpha \in AS$). The letter A in the names of the axioms in table 5 refers to *absolute time* (versions with relative time were also considered in [BAB91a], but are not treated here).

2.5 Algebra of Communicating Processes.

An axiomatization of parallel composition with communication uses the left merge operator $\between$, the communication merge operator $|$, and the encapsulation operator ∂_H of [BK84]. Moreover, two extra auxiliary operators introduced in [BAB91a] are needed: the ultimate delay operator and the bounded initialization operator.

The ultimate delay operator U takes a process expression X, and returns an element of $\mathbb{R}^{\geq 0}$. The intended meaning is that X can idle before $U(X)$, but X can never reach time $U(X)$ or a later time by just idling.

$X \between Y = X \between Y + Y \between X + X \mid Y$	CM1	$U(\alpha(t)) = t$	ATU1
$\alpha(t) \between X = (\alpha(t) \gg U(X)) \cdot X$	ATCM2	$U(\delta(t)) = t$	ATU2
$(\alpha(t) \cdot X) \between Y = (\alpha(t) \gg U(Y)) \cdot (X \between Y)$	ATCM3	$U(X + Y) = \max\{U(X), U(Y)\}$	ATU3
$(X + Y) \between Z = X \between Z + Y \between Z$	CM4	$U(X \cdot Y) = U(X)$	ATU4
$(\alpha(t) \cdot X) \mid \beta(r) = (\alpha(t) \mid \beta(r)) \cdot X$	CM5'		
$\alpha(t) \mid (\beta(r) \cdot X) = (\alpha(t) \mid \beta(r)) \cdot X$	CM6'	$r \geq t \Rightarrow \alpha(r) \gg t = \delta(t)$	ATB5
$(\alpha(t) \cdot X) \mid (\beta(r) \cdot Y) = (\alpha(t) \mid \beta(r)) \cdot (X \between Y)$	CM7'	$r < t \Rightarrow \alpha(r) \gg t = \alpha(r)$	ATB6
$(X + Y) \mid Z = X \mid Z + Y \mid Z$	CM8	$(X + Y) \gg t = (X \gg t) + (Y \gg t)$	ATB7
$X \mid (Y + Z) = X \mid Y + X \mid Z$	CM9	$(X \cdot Y) \gg t = (X \gg t) \cdot Y$	ATB8
$\partial_H(a) = a$ if $a \notin H$	D1		
$\partial_H(a) = \delta$ if $a \in H$	D2		
$\partial_H(\alpha \,\&\, \beta) = \partial_H(\alpha) \,\&\, \partial_H(\beta)$	ASD		
$\partial_H(\alpha(t)) = (\partial_H(\alpha))(t)$	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 6. Remaining axioms of ACPρ.

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 BPA$\rho\delta$. With $X \gg t$ we denote the process X with its behaviour restricted to the extent that its first action must be performed at a time before $t \in \mathbb{R}^{\geq 0}$.

The axioms of ACPρ are in tables 1 through 6. In table 6, $H \subseteq A$, $\alpha,\beta \in AS$, $a \in A_\delta$.

2.6 Operational semantics.

We describe an operational semantics for ACPρ following Klusener [K91] and [BAB91c], a reformulisation of the original semantics in [BAB91a]. We have a binary relation $\xrightarrow{\mu}$ and a unary relation $\xrightarrow{\mu}\surd$ on closed process expressions for each $\mu \in AST$. In case $\mu \neq \delta(t)$, the extension of the relations is found as the least fixed point of a simultaneous inductive definition.

$$r > 0 \Rightarrow \alpha(r) \xrightarrow{\alpha(r)} \surd$$

$$\frac{x \xrightarrow{\alpha(r)} x'}{x+y \xrightarrow{\alpha(r)} x',\ y+x \xrightarrow{\alpha(r)} x'} \qquad \frac{x \xrightarrow{\alpha(r)} \surd}{x+y \xrightarrow{\alpha(r)} \surd,\ y+x \xrightarrow{\alpha(r)} \surd}$$

$$\frac{x \xrightarrow{\alpha(r)} x'}{x \cdot y \xrightarrow{\alpha(r)} x' \cdot y} \qquad \frac{x \xrightarrow{\alpha(r)} \surd}{x \cdot y \xrightarrow{\alpha(r)} r \gg y}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ r > s}{s \gg x \xrightarrow{\alpha(r)} x'} \qquad \frac{x \xrightarrow{\alpha(r)} \surd,\ r > s}{s \gg x \xrightarrow{\alpha(r)} \surd}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ r < U(y)}{x \| y \xrightarrow{\alpha(r)} x' \| (r \gg y),\ x \between y \xrightarrow{\alpha(r)} x' \| (r \gg y),\ y \| x \xrightarrow{\alpha(r)} (r \gg y) \| x'}$$

$$\frac{x \xrightarrow{\alpha(r)} \surd,\ r < U(y)}{x \| y \xrightarrow{\alpha(r)} r \gg y,\ x \between y \xrightarrow{\alpha(r)} r \gg y,\ y \| x \xrightarrow{\alpha(r)} r \gg y}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ y \xrightarrow{\beta(r)} y',\ \alpha \mid \beta = \gamma \neq \delta}{x \| y \xrightarrow{\gamma(r)} x' \| y',\ x \mid y \xrightarrow{\gamma(r)} x' \| y'} \qquad \frac{x \xrightarrow{\alpha(r)} \surd,\ y \xrightarrow{\beta(r)} \surd,\ \alpha \mid \beta = \gamma \neq \delta}{x \| y \xrightarrow{\gamma(r)} \surd,\ x \mid y \xrightarrow{\gamma(r)} \surd}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ y \xrightarrow{\beta(r)} \surd,\ \alpha \mid \beta = \gamma \neq \delta}{x \| y \xrightarrow{\gamma(r)} x',\ y \| x \xrightarrow{\gamma(r)} x',\ x \mid y \xrightarrow{\gamma(r)} x',\ y \mid x \xrightarrow{\gamma(r)} x'}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ r < t}{x \gg t \xrightarrow{\alpha(r)} x'} \qquad \frac{x \xrightarrow{\alpha(r)} \surd,\ r < t}{x \gg t \xrightarrow{\alpha(r)} \surd}$$

$$\frac{x \xrightarrow{\alpha(r)} x',\ ats(\alpha) \cap H = \emptyset}{\partial_H(x) \xrightarrow{\alpha(r)} \partial_H(x')} \qquad \frac{x \xrightarrow{\alpha(r)} \surd,\ ats(\alpha) \cap H = \emptyset}{\partial_H(x) \xrightarrow{\alpha(r)} \surd}$$

Table 7. Action rules for non-δ actions for ACPρ.

The inductive rules for the operational semantics are similar to those used in structured operational semantics. We list the rules for non-δ multi-actions. In table 7, we have $\alpha,\beta,\gamma \in AS - \{\delta\}$, $r > 0$ (we never allow timestamp 0!), $s,t \geq 0$, x,x',y,y' are closed process expressions. $ats(\alpha)$ is the set of atomic actions occurring in the multi-action α.

2.7 DELTA-TRANSITIONS.

Now we construct a transition system for a term as follows: first generate all transitions involving non-δ actions using the inductive rules of table 7. Then, for every node (term) p in this transition system we do the following: first determine its ultimate delay $U(p) = u$ by means of table 5 and the additional axioms below. Then, if the ultimate delay is larger than the supremum of the time stamps of all outgoing transitions, we add a transition

$$p \xrightarrow{\delta(u)} \surd.$$

Otherwise, we do nothing (add no transitions).

We see that the action rules for parallel composition, and the determination of δ-transitions, make use of the ultimate delay operator. We add the axioms in table 8 so that the ultimate delay can be syntactically determined for every closed term.

$U(t \gg X) = \max\{t, U(X)\}$	ATU5
$U(X \parallel Y) = \min\{U(X), U(Y)\}$	ATU6
$U(X \between Y) = \min\{U(X), U(Y)\}$	ATU7
$U(X \mid Y) = \min\{U(X), U(Y)\}$	ATU8
$U(X \gg t) = \min\{t, U(X)\}$	ATU9
$U(\partial_H(X)) = U(X)$	ATU10

Table 8. Ultimate delay axioms.

2.8 BISIMULATIONS.

A *bisimulation* on is a binary relation R on closed process expressions such that ($\mu \in AST$):

i. for each p and q with $R(p, q)$: if there is a step μ possible from p to p', then there is a closed process expression q' such that $R(p', q')$ and there is a step μ possible from q to q'.

ii. for each p and q with $R(p, q)$: if there is a step μ possible from q to q', then there is a CPE p' such that $R(p', q')$ and there is a step μ possible from p to p'.

iii. for each p and q with $R(p, q)$: a termination step μ to $\surd$ is possible from p iff it is possible from q.

We say expressions p and q are *bisimilar*, denoted $p \leftrightarrow q$, if there exists a bisimulation on closed process expressions with $R(p,q)$. In [K91] it is shown that bisimulation is a congruence relation on closed process expressions, and that closed process expression modulo bisimulation determine a model for ACPρ. Indeed, this model is isomorphic to the initial algebra. The advantage of this operational semantics is, that it allows extensions to models containing recursively defined processes.

It is also possible to give an explicit graph model for ACPρ, see [BAB91c].

2.9 INTEGRATION.

An extension of ACPρ (called ACPρI) that is very useful in applications is the extension with the integral operator, denoting a choice over a continuum of alternatives. I.e., if V is a subset of $\mathbb{R}^{\geq 0}$, and v is a variable over $\mathbb{R}^{\geq 0}$, then $\int_{v \in V} P$ denotes the alternative composition of alternatives P[t/v] for $t \in V$ (expression P with nonnegative real t substituted for variable v). For more information, we refer the reader to [BAB91a] and [K91]. The operational semantics is straightforward (table 9, $\mu \in AST$).

$\dfrac{x(t) \xrightarrow{\mu} x',\ t \in V}{\int_{v \in V} x(v) \xrightarrow{\mu} x'}$	$\dfrac{x(t) \xrightarrow{\mu} \surd,\ t \in V}{\int_{v \in V} x(v) \xrightarrow{\mu} \surd}$

Table 9. Action relations for integration.

We will not provide axioms for the integral operator here (and refer the reader to [BAB91a] and [K91]), except for the axiom for the ultimate delay operator:

$$U(\int_{v \in V} P) = \sup\{U(P[t/v]) : t \in V\} \qquad \text{ATU11.}$$

3. ASYNCHRONOUS COMMUNICATION.

3.1 DEFINITION.

Let C be a finite collection of port names, and let D be a finite set of data. As special symbolic atomic actions we introduce, following [BEKT85], for $c \in C$, $d \in D$:

- $c\uparrow d$ potential send of message d along asynchronous channel c
- $c\Uparrow d$ effectuated send of message d along asynchronous channel c
- $c\downarrow d$ potential receive of message d along asynchronous channel c
- $c\Downarrow d$ effectuated receive of message d along asynchronous channel c.

We define $c\Downarrow D = \{c\Downarrow d \mid d \in D\}$, and likewise for the other actions.

Besides these asynchronous communication actions, we have the standard synchronous communication actions, for $k \in H$, $d \in D$:

- $s_k(d)$ send message d at synchronous port k
- $r_k(d)$ receive message d at synchronous port k
- $c_k(d)$ communicate message d at synchronous port k.

We define $s_rHD = \{s_k(d), r_k(d) \mid k \in H, d \in D\}$, $cHD = \{c_k(d) \mid k \in H, d \in D\}$.

3.2. STATE OPERATOR.

The state operator was introduced in [BAB88]. It keeps track of the global state of a system, and is used to describe actions that have a side effect on a state space. In [BAB91c], it was shown how to define a state operator on real time process algebra with finitely many locations. There, an ordering on locations was needed in order to get a right definition for multi-actions. Here, we define a specific state operator, on a domain with infinitely many locations, that does not require this ordering.

The state operator comes equipped with two functions: given a certain state and an action to be executed from that state, the function action gives the resulting action and the function effect the resulting state. Different from In [BAB91c], here it will be needed that the action and effect function also depend on the place and time of an action. Hence, these functions are not defined on A, but on AST.

For each asynchronous channel $c \in C$, we will have a state operator λ^c. The set of states $\mathbb{V}$ is the collection of all finite subsets of $D \times \mathbb{R}^3 \times \mathbb{R}^{\geq 0}$. Moreover, v is a given constant.

The state operator λ^c_V ($V \in \mathbb{V}$) has functions

$\text{action}_c: \text{AST} \times \mathbb{V} \to \text{AST}$ $\quad$ $\text{effect}_c: \text{AST} \times \mathbb{V} \to \mathbb{V}$

given by:

$\text{action}_c(c\uparrow d(x,t), V) = c\Uparrow d(x,t)$

$\text{effect}_c(c\uparrow d(x,t), V) = V \cup \{\langle d, x, t\rangle\}$

$\text{action}_c(c\downarrow d(y, r), V) = c\Downarrow d(y, r)$ $\quad$ if there is $\langle d, x, t\rangle \in V$ such that $|y - x| = v\cdot(r - t)$ and $r \neq t$

$\text{action}_c(c\downarrow d(y, r), V) = \delta(r)$ $\quad$ otherwise

$\text{effect}_c(c\downarrow d(y, r), V) = V$

$\text{action}_c(a(x,t), V) = a(x,t)$ $\quad$ for all a not of the form $c\uparrow d$ or $c\downarrow d$

$\text{effect}_c(a(x,t), V) = V$ $\quad$ for all a not of the form $c\uparrow d$ or $c\downarrow d$.

Now these functions are extended to multi-actions in the obvious way:

$\text{action}_c((a(x) \,\&\, \alpha)(t), V) = \text{action}_c(a(x,t), V) \,\&\, \text{action}_c(\alpha(t), V)$

$\text{effect}_c((a(x) \,\&\, \alpha)(t), V) = \text{effect}_c(a(x,t), \text{effect}_c(\alpha(t), V))$.

3.3. LEMMA. Let $c \in C$, $V \in \mathbb{V}$, $\mu, \nu \in \text{AST}$. Then $\text{effect}_c(\mu \,\&\, \nu, V) = \text{effect}_c(\nu \,\&\, \mu, V)$.

3.4 DEFINITION.

The defining equations for the state operator are now straightforward (cf. [BAB88]). In table 10, $V \in \mathbb{V}$, $c \in C$, $\mu \in \text{AST}$, x,y processes.

$\lambda^c_V(\mu) = \text{action}(\mu, V)$	SO1
$\lambda^c_V(\mu\cdot x) = \text{action}(\mu, V)\cdot\lambda^c_{\text{effect}(\mu,V)}(x)$	SO2
$\lambda^c_V(x + y) = \lambda^c_V(x) + \lambda^c_V(y)$	SO3
$\lambda^c_V(\int_{t\in T} P) = \int_{t\in T} \lambda^c_V(P)$	SO4

Table 10. State operator.

It is equally straightforward to give action rules for the operational semantics ($\mu,\nu \in AST, \mu,\nu \neq \delta(r)$).

$$\frac{x \xrightarrow{\mu} x',\ action_c(\mu,V)=\nu\neq\delta(r)}{\lambda_V^c(x) \xrightarrow{\nu} \lambda^c_{effect_c(\mu,V)}(x')} \qquad \frac{x \xrightarrow{\mu} \surd,\ action_c(\mu,V)=\nu\neq\delta(r)}{\lambda_V^c(x) \xrightarrow{\nu} \surd}$$

Table 11. Action relations for the state operator.

In order to deal with δ-transitions, we add the axiom

$$U(\lambda_V^c(x)) = U(x) \qquad \text{ATU12.}$$

Then, we determine the existence of δ-transitions as before.

3.5 EXAMPLE.

Let us consider two processes S and R at fixed locations, communicating through an asynchronous channel 2 (see fig. 1). 1 and 3 are synchronous ports. w_0 and w_1 are system delay constants. Suppose the distance between S and R is equal to the distance a message travels in 1 time unit.

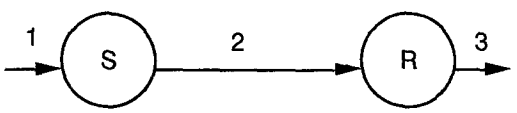

FIGURE 1.

We have the following specifications:

$$S(r) = \int_{t>r} \sum_{d\in D} r1(d)(x,t)\cdot 2\uparrow d(x, t+w_0)\cdot S(t+2w_0) \qquad (r \geq 0)$$

$$R(r) = \int_{t>r} \sum_{d\in D} 2\downarrow d(y,t)\cdot s3(d)(y, t+w_1)\cdot R(t+2w_1) \qquad (r \geq 0).$$

The system is now described by the expression $\lambda^2_\emptyset(S(0) \parallel R(0))$.

After a certain $2\uparrow d(x, t_0)$-action at time t_0, the triple $\langle d, x, t_0\rangle$ is added to the state. Then, action $2\downarrow d(y, t_0+1)$ is transformed into $2\Downarrow d(y, t_0+1)$, and all other $2\downarrow d(y, t)$ for $t \neq t_0+1$ are renamed to $\delta(t)$. However, the presence (in a sum context) of $\delta(t)$ with $t > t_0+1$ allows R to bypass the option $2\Downarrow d(y, t_0+1)$, wait too long before trying to receive, and then it is too late, and no further action will be possible. Thus, we have to enforce that the action $2\Downarrow d$ occurs as soon as it is possible.

This is a kind of *maximal progress* or *maximal liveness* assumption (cf. [RR88]). We will ensure that communication takes place as soon as possible by use of the priority operator of [BABK86]. A similar operator was used in JEFFREY [J91c].

4. PRIORITIES.

Let us start by recalling the priority axioms from [BABK86]. There, an operator θ is defined, that cancels out all actions in a sum context, that do not have maximal priority. Priorities are given by a partial ordering on actions. Here, we have a very simple ordering: we have a certain set $H \subseteq A$, and all actions from H have priority over actions from outside H. In example 3.5, H is the set $2\Downarrow D$.

4.1 DEFINITION.
Let a set $H \subseteq A$ be given. The *priority operator* θ_H has the axioms in table 12 ($\alpha \in AS$).

$\theta_H(\alpha(t)) = \alpha(t)$	TH1
$\theta_H(x + y) = \theta_H(x) \lhd_H y + \theta_H(y) \lhd_H x$	TH2
$\theta_H(\alpha(t) \cdot x) = \alpha(t) \cdot \theta_H(t \gg x)$	TH3
$\theta_H(\int_{v \in V} P) = \int_{v \in V} (\theta_H(P) \lhd_H \int_{t \in V - \{v\}} P[t/v])$	TH4

Table 12. Priority operator.

The priority operator is axiomatised by means of the *unless operator* $\lhd_H$. In $x \lhd_H y$, a starting action from H in y will cancel all starting actions in x with a later timestamp, and all starting actions not from H with the same timestamp. The unless operator can be axiomatised directly, but we will not do so here. Instead, we will define two other operators (one of which is a generalisation of the bounded initialisation operator) that are useful in their own right, and allow to define the unless operator easily.

4.2 MINIMAL DELAY WITH PRIORITY.
The operator D_H gives the minimal delay with priority, i.e. $D_H(x)$ determines the infimum of all times at which x can perform an initial action in H. If x can perform no initial action in H, we will set $D_H(x) = \infty$. In table 13, $H \subseteq A$, $a \in A$, $\alpha, \beta \in AS$, $x, y \in P$, X a process with time variable t.

$D_H(\delta(t)) = \infty$		MDP1
$D_H(a(t)) = \infty$	if $a \notin H$	MDP2
$D_H(a(t)) = t$	if $a \in H$	MDP3
$D_H((\alpha \,\&\, \beta)(t)) = \min\{D_H(\alpha(t)), D_H(\beta(t))\}$		MDP4
$D_H(x + y) = \min\{D_H(x), D_H(y)\}$		MDP5
$D_H(x \cdot y) = D_H(x)$		MDP6
$D_H(\int_{t \in T} X) = \inf\{D_H(X) \mid t \in T\}$		MDP7

Table 13. Minimal delay with priority.

4.3 BOUNDED INITIALISATION WITH PRIORITY.
Next, we define the operator $\gg_H$. This operator is just like the bounded initialisation operator of 2.5, but if the timestamp left coincides with the time right, actions from H will survive, but actions outside H will not. In table 14, $H \subseteq A$, $r \geq 0$, $a \in A$, $\alpha, \beta \in AS$, $x, y \in P$, X a process with time variable t. We see that the bounded initialisation operator $\gg$ is just $\gg_\varnothing$.

$x \gg_H \infty = x$		BIP1
$a(t) \gg_H r = a(t)$	if $t < r$ or $t = r$ and $a \in H$	BIP2
$a(t) \gg_H r = \delta(r)$	if $t > r$ or $t = r$ and $a \notin H$	BIP3
$(\alpha \,\&\, \beta)(t) \gg_H r = (\alpha(t) \gg_H r) \,\&\, (\beta(t) \gg_H r)$		BIP4
$(x + y) \gg_H r = (x \gg_H r) + (y \gg_H r)$		BIP5
$(x \cdot y) \gg_H r = (x \gg_H r) \cdot y$		BIP6
$(\int_{t \in T} X) \gg_H r = \int_{t \in T} (X \gg_H r)$		BIP7

Table 14. Bounded initialisation with priority.

4.4 UNLESS OPERATOR.

Now with the use of these minimal delay and bounded initialisation operators with priority the unless operator can be defined easily:

$$x \lhd_H y = x \gg_H D_H(y).$$

The typical axioms for the unless operator (see [BABK86]) can now be derived.

Returning to example 3.5, we see that the correct expression for the system is as follows:

$$\theta_2 \Downarrow_D \circ \hat{\lambda}^2_{\mathcal{D}}(S(0) \parallel R(0)).$$

4.5 EXAMPLE.

A consequence of axiom TH4 is that we have the following identity: $\theta_H(\int_{t\in(1,2)} h(t)) = \delta(1)$, if $h \in H$. This is because each h action is canceled by one with lesser timestamp. We can call this phenomenon a *priority deadlock*. The equation can be generalized to the following: $\theta_H(x + \int_{t\in(1,2)} h(t)) = \theta_H(x) \gg_H 1$ for all x.

4.6 EXAMPLE.

We now discuss a larger example, a protocol transmitting data via a mobile intermediate station.

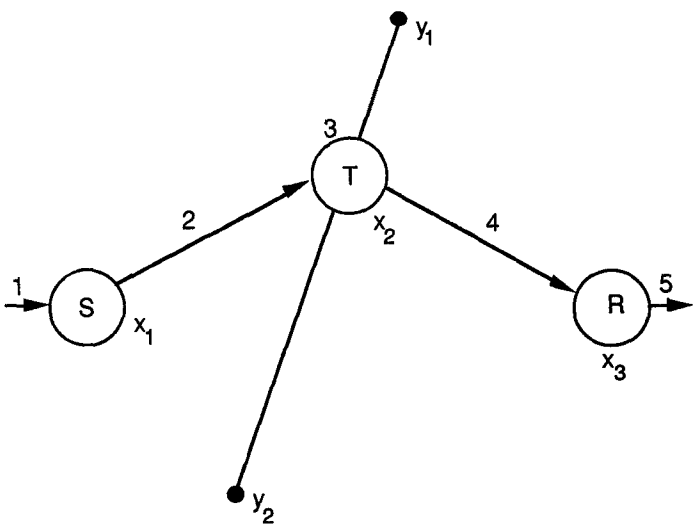

FIGURE 2.

In figure 2, we have a sender S at location x_1, a receiver R at location x_3, and a transmitter T that is moving on a line between locations y_1 and y_2, starting at y_1 at time t=0. The location of the transmitter is given by the following formula:

$$x_2(t) = y_1 + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega t)) \cdot (y_2 - y_1).$$

The transmitter consists of two parts, a receiving part and a sending part. These parts are interconnected by a synchronous communication port 3. 1 and 5 are also synchronous ports, and 2 and 4 are asynchronous channels. We have system delay constants w_1, w_2, w_3, w_4.

The components are now given by the following specification:

$$S(r) = \int_{t\geq r} \sum_{d\in D} r1(d)(x_1, t)\cdot 2\uparrow d(x_1, t+w_1) \cdot S(t + 2w_1) \qquad (r\geq 0)$$

$$T_R(r) = \int_{t\geq r} \sum_{d\in D} 2\downarrow d(x_2(t), t)\cdot s3(d)(x_2(t+w_2), t+w_2) \cdot T_R(t + 2w_2) \qquad (r\geq 0)$$

$$T_S(r) = \int_{t\geq r} \sum_{d\in D} r3(d)(x_2(t), t)\cdot 4\uparrow d(x_2(t+w_3), t+w_3) \cdot T_S(t + 2w_3) \qquad (r\geq 0)$$

$$R(r) = \int_{t\geq r} \sum_{d\in D} 4\downarrow d(x_3, t)\cdot s5(d)(x_3, t+w_4) \cdot S(t + 2w_4) \qquad (r\geq 0).$$

Now the system is given by the identity:

$$SYS = \partial_{sr3D}\big(\theta_{2\Downarrow D}\circ\lambda^2_{\varnothing}(S(0) \parallel T_R(0)) \parallel \theta_{4\Downarrow D}\circ\lambda^4_{\varnothing}(T_S(0) \parallel R(0))\big).$$

The protocol works well if the data enter at a sufficiently low frequency.

4.7 THE REPLACEMENT OPERATOR: STRUCTURED NOTATION FOR A MOBILE PROCESS.

Example 4.6 demonstrates a process T that is mobile in the sense that its actions take place at locations that vary in time. In this section, we will have a closer look at the description of such mobile processes.

Let $f: \mathbb{R}^3 \times \mathbb{R}^{\geq 0} \to \mathbb{R}^3$ be a continuous function. The operator ρ_f modifies each action $a(x, t)$ of a process P into $a(f(x,t), t)$. Thus, at time t, all spatial coordinates are translated according to the mapping $f_t: x \mapsto f(x, t)$. The equations for ρ_f are very simple (table 15).

$\rho_f(\delta(t)) = \delta(t)$	RRN1
$\rho_f(a(x, t)) = a(f(x,t), t)$	RRN2
$\rho_f((\alpha \,\&\, \beta)(t)) = \rho_f(\alpha(t)) \,\&\, \rho_f(\beta(t))$	RRN3
$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$	RRN4
$\rho_f(x\cdot y) = \rho_f(x)\cdot\rho_f(y)$	RRN5
$\rho_f(\int_{t\in T} X) = \int_{t\in T} \rho_f(X)$	RRN6

Table 15. Replacement operator.

ρ_f is a renaming but in fact it affects spatial coordinates only, and not action labels. We prefer to call ρ_f a *replacement operator*.

In example 4.6, we can rephrase the definition of T as follows: let x be a point in space. Then we can define

$$f(x, t) = x + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega t))\cdot(y_2 - y_1).$$

Now put

$$T^*_R(x, r) = \int_{t\geq r} \sum_{d\in D} 2\downarrow d(x, t)\cdot s3(d)(x, t+w_2)\cdot T_R(t + 2w_2),$$

$$T^*_S(x, r) = \int_{t\geq r} \sum_{d\in D} r3(d)(x, t)\cdot 4\uparrow d(x, t+w_3)\cdot T_S(t + 2w_3).$$

Now the processes in 4.6 are given by: $T_R(t) = \rho_f(T^*_R(y_1, t))$, $T_S(t) = \rho_f(T^*_S(y_1, t))$.

Using this notation we obtain a better modular structure. The system specification becomes:

$$SYS = \partial_{sr3D}\big(\theta_{2\Downarrow D}\circ\lambda^2_{\varnothing}(S(0) \parallel \rho_f(T^*_R(y_1, 0))) \parallel \theta_{4\Downarrow D}\circ\lambda^4_{\varnothing}(\rho_f(T^*_S(y_1,0)) \parallel R(0))\big).$$

Now suppose that the intermediate process T oscillates in a direction perpendicular to the segment (y_1, y_2) as well: let two points z_1, z_2 be given such that $(y_1, y_2) \perp (z_1, z_2)$. Define

$$g(x, t) = x + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega^* t))\cdot(z_2 - z_1),$$

then the combined motion of T is given by the expressions $\rho_g \circ \rho_f(T^*_R(y_1, 0))$, $\rho_g \circ \rho_f(T^*_S(y_1, 0))$.

Thus, superposition of motions corresponds to composition of the corresponding replacement operators.

4.8 SIGNAL INTENSITY.

We proceed by discussing some variations in the communication mechanism. First of all, suppose we want to describe asynchronous communication with a decrease in signal intensity.

Assume that data are sent with intensity i_0 and that intensity decreases proportionally to the square of the distance traveled. So if i_0 is the signal intensity at distance 1 from the source, the signal intensity at distance is $a\cdot i_0\cdot^{-2}$ for some constant $a \geq 0$. Let the receiving actions be equipped with an additional real parameter j that indicates at which intensity a signal can be received. Then the action function will allow $c\downarrow(d, j)(y, r)$ provided there exist x,t such that d was sent at (x, t) and

i. $|y - x| = v\cdot(r - t)$

ii. $a\cdot i_0\cdot |y - x|^{-2} = j$.

A receiver that allows to receive messages between intensities lo, hi and between times fi, la will show integration over the intensity interval:

$$\text{Receiver} = \int_{j \in [lo, hi]} \int_{t \in [fi, la]} \sum_{d \in D} c\downarrow(d, j)(y(t), t)\cdot P(d, t).$$

4.9 IMPENETRABLE OBJECTS.

Next, we look at how to model a signal transmission space with an impenetrable object. Let B(2, z) be the closed ball with radius 2 and center z. In this case, the action function works as follows:

$\text{action}_c(c\uparrow d(x,t), V) = \delta(t)$ if $x \in B(2, z)$ (i.e. if $|x - z| \leq 2$)

$\text{action}_c(c\uparrow d(x,t), V) = c\uparrow d(x,t)$ if $x \notin B(2, z)$

$\text{effect}_c(c\uparrow d(x,t), V) = V$ if $x \in B(2, z)$

$\text{effect}_c(c\uparrow d(x,t), V) = V \cup \{\langle d,x,t\rangle\}$ if $x \notin B(2, z)$

$\text{action}_c(c\downarrow(d,j)(y,r), V) = c\Downarrow d(y,r)$ if there is $\langle d, x, t\rangle \in V$ such that $|y - x| = v\cdot(r - t)$ and $a\cdot i_0\cdot |y - x|^{-2} = j$ and the line segment from y to x does not intersect B(2, z)

$\text{action}_c(c\downarrow(d,j)(y,r), V) = \delta(r)$ otherwise.

As usual, $c\downarrow d$ actions have no effect on the state, so the effect function does not change the state.

4.10 REFLECTION.

A more complex example is obtained if one allows reflection of the signal. Carrying on with the previous example, we allow the signal to be reflected at the surface of the ball B(2, z). We assume perfect reflection without loss of signal intensity. Like before, we allow that a signal (or various signals) is simultaneously delivered at (y,r) through different paths. This can happen even with one source. It is possible that a signal d is sent by executing $c\uparrow d(x,t)$ and it reaches (y,r) along two different paths, both containing a reflection. In the case of the ball B(2, z) this is not possible of course, but with two balls this could happen indeed. The action function remains the same in the case of $c\uparrow d(x,t)$ but for $c\downarrow(d,j)(y,r)$ we get:

$action_c(c{\downarrow}(d,j)(y,r), V) = c{\Downarrow}d(y,r)$
if (1) there is $\langle d, x, t\rangle \in V$ such that $|y - x| = v\cdot(r - t)$ and $a\cdot i_0\cdot |y - x|^{-2} = j$ and the line segment from y to x does not intersect $B(2, z)$;
or (2) there is $\langle d, x, t\rangle \in V$ and w on the boundary of $B(2, z)$ such that $|y - w| + |w - x| = v\cdot(r - t)$, $a\cdot i_0\cdot(|y - w| + |w - x|)^{-2} = j$, x,y,z,w are in one plane, and (z,w) bisects the angle (x,w,y);
$action_c(c{\downarrow}(d,j)(y,r), V) = \delta(r)$ otherwise.

A state operator based on this action function describes an environment in which asynchronous communication through a non-trivial physical medium is supported. We remark that if the sending process also receives reflected signals, we are dealing with remote sensing, rather than asynchronous communication.

Interesting complications arise if we intend to model an asynchronous communication where, firstly, there are several independently moving objects that are impenetrable for signals but that do allow reflections with varying reflection coefficients, and, secondly, these objects have their motion guided by actions (in a discontinuous way) and by laws of classical mechanics between actions. This is in fact what happens if a sensor is used to assist a robot in navigating through a number of independently moving objects, each having its own surface characteristics.

It is our impression that the language of classical real space process algebra with priority and state operators is not sufficient to model the sensors needed for robot navigation in a dynamic environment.

5. ASYNCHRONOUS MESSAGE PASSING EXPRESSED IN TERMS OF SYNCHRONOUS COMMUNICATION.

In many cases, asynchronous message passing can be expressed using the available synchronous communication mechanism of ACPρ. In this approach, which appears in many forms throughout the literature, a process is introduced that represents the medium through which the data are being transported.

In this section we will provide various examples of such processes, representing an asynchronous transport medium. It turns out that a real time version of the process creation mechanism of [BE90] is useful to define a variety of such processes in a uniform way. For that purpose we describe process creation first, a mechanism which is of independent interest also.

It should be noted that the use of the state operator in the previous sections has been motivated by our inability to describe the particular form of asynchronous message transfer of example 4.6 in the more traditional fashion of this section. This in no way excludes that an elegant description of example 4.6 based on synchronous communication is possible, but our search has been without success.

5.1 PROCESS CREATION.

For the description of process creation, we assume that we have special disjoint subsets $cr(D) = \{cr(d) \mid d \in D\}$ and $\overline{cr}(D) = \{\overline{cr}(d) \mid d \in D\}$ within the set of symbolic atomic actions A, with $cr(d) \mid a = \overline{cr}(d) \mid a = \delta$ for all $d \in D$ and $a \in A_\delta$.

Further, we assume the existence of a function ϕ: $D \times \mathbb{R}^3 \times \mathbb{R}^{\geq 0} \to P$, and we require ϕ to be defined by means of guarded recursion equations. Like in [BE90], the function ϕ determines a process to be created from action cr(d). In the real time case, $\phi(d, x, t)$ represents the process created from the timed action $cr(d)(x, t)$.

Next we use, as in the symbolic case, an operator E_ϕ which enforces process creation from cr(d) actions occurring in its scope. The equations of E_ϕ are in table 16. In order to state these axioms, we

need some extra notation. We need to be able to determine the set $CR(\alpha) = \{\langle d, x\rangle \in D \times \mathbb{R}^3 \mid cr(d)$ occurs in $\alpha\}$ for a multi-action α. This set can be defined recursively as follows:

$$CR(a(x) \,\&\, \alpha) = CR(\alpha) \qquad \text{if } a \notin cr(D)$$
$$CR(cr(d)(x) \,\&\, \alpha) = \{\langle d, x\rangle\} \cup CR(\alpha).$$

Also, we need the notation $\overline{\alpha}$ for the multi-action α, where all $cr(d)$ actions are changed into $\overline{cr}(d)$ actions, i.e.

$$\overline{a(x) \,\&\, \alpha} = a(x) \,\&\, \overline{\alpha} \qquad \text{if } a \notin cr(D)$$
$$\overline{cr(d)(x) \,\&\, \alpha} = \overline{cr}(d)(x) \,\&\, \overline{\alpha}.$$

$E_\phi(\alpha(t)) = \alpha(t)$	if $CR(\alpha) = \varnothing$	PCT1
$E_\phi(\alpha(t)) = \overline{\alpha}(t) \cdot E_\phi(\parallel_{\langle d,x\rangle \in CR(\alpha)} \phi(d,x,t))$	otherwise	PCT2
$E_\phi(\alpha(t) \cdot X) = \alpha(t) \cdot E\phi(X)$	if $CR(\alpha) = \varnothing$	PCT3
$E_\phi(\alpha(t) \cdot X) = \overline{\alpha}(t) \cdot E_\phi(X \parallel \parallel_{\langle d,x\rangle \in CR(\alpha)} \phi(d,x,t))$	otherwise	PCT4
$E_\phi(x + y) = E_\phi(x) + E_\phi(y)\}$		PCT5
$E_\phi(\int_{t\in T} X) = \int_{t\in T} E_\phi(X)$		PCT6

Table 16. Process creation.

The operational semantics of the process creation operator now follows easily on the basis of the axioms. Note that the process creation mechanism can be used in real time (without space) just as well. All examples of [BE90] can be adapted to a real time setting. We concentrate here on examples concerning message transport media.

5.2 MESSAGE HANDLER.

The following recursive specification defines a process that, upon receiving an input, creates a message handler to take care of the input.

$$M(r) = \int_{t>r+t_0} \sum_{d\in D} r(d)(x(t), t) \cdot cr(d)(x(t+t_1), t+t_1) \cdot M(t+t_1) \qquad \text{(for } r>0\text{)}.$$

Here t_0 is a parameter that determines a minimum delay between the creation of the message handler and reception of a new input, whereas t_1 determines the delay between reception of a datum and creation of the corresponding handler. $x(t)$ is some function from $\mathbb{R}^{\geq 0}$ to $\mathbb{R}^3$ that determines the input location of the medium. We assume that the handler is created at that very same logical location but with a delay of t_1 (i.e. at $x(t + t_1)$).

Next, we provide possible functions ϕ_n: $D \times \mathbb{R}^3 \times \mathbb{R}^{\geq 0} \to P$ that determine the handler created. This leads to an asynchronous message transport medium $M_n = E_{\phi_n}(M)$ for each case. In all cases, $y(x, r)$ is the output location of the medium at time $t + r$ if at time t the input location is x. Thus, input at (x, t) leads to output at $(y(x, \Delta), t+\Delta)$ for some $\Delta > 0$.

CASE 1: $\phi_1(d, x, t) = s(d)(y(x, \Delta_1), t+\Delta_1)$.

In this case data need a constant time Δ_1 to travel through the medium. Moreover, the medium introduces no errors or omissions.

CASE 2: $\phi_2(d, x, t) = s(d)(y(x, \Delta_2(t)), t+\Delta_2(t))$.

In this case, the transmission time depends on t. This happens e.g. if the output location is moving relative to the input location. Notice that the difference with example 4.6 is that there, due to the

broadcasting nature of the mechanism, the output location is not known in the same way. In particular, the function $\Delta_2(t)$ is not easy to define in that example, even if one assumes that broadcasting is irrelevant and each message is received exactly once. This is caused by the fact that in example 4.6 the output location (i.e. location (x, t) at which an action $c\Downarrow d(x, t)$ happens) must be derived from the combined behavior of *two* processes.

CASE 3: $\phi_3(d, x, t) = \int_{r\in(\Delta_1-\varepsilon_1,\Delta_1+\varepsilon_1)} s(d)(y(x, r-t), r).$

M_3 is like M_1, be it that there is a tolerance ε_1 in the arrival time. If $\varepsilon_1 > \frac{1}{2}(t_0 + t_1)$, complications can arise because M_3 may deliver different messages at the same time and place. Assuming $\varepsilon_1 < \frac{1}{2}(t_0 + t_1)$, this unwanted interference is not possible.

CASE 4: $\phi_4(d, x, t) = \int_{r\in(\Delta_2(t)-\varepsilon_1,\Delta_2(t)+\varepsilon_1)} s(d)(y(x, r-t), r).$

This example adds an arrival time tolerance to example 2.

Next, we assume that the orbit of a message d traveling from (x,t) to $(y(x,\Delta), t+\Delta)$ is given by $(z(r-t), r)$, so in particular, we have $(x,t) = (z(0), t+0)$ and $(y(x,\Delta), t+\Delta) = (z(\Delta), t+\Delta)$.

CASE 5: $\phi_5(d, x, t) = s(d)(y(x, \Delta_1), t+\Delta_1) + \int_{r\in(0,\Delta_1)} lost(z(r), t+r) \cdot s(\perp)(y(x,\Delta_1), t+\Delta_1).$

This example allows the datum to be changed into $\perp$ anywhere during transmission. In particular, M_5 keeps track where a datum gets lost. In the next example, M_6 will not deliver lost data.

CASE 6: $\phi_6(d, x, t) = s(d)(y(x, \Delta_1), t+\Delta_1) + \int_{r\in(0,\Delta_1)} lost(z(r), t+r).$

CASE 7: $\phi_7(d, x, t) = s(d)(y(x, \Delta_1), t+\Delta_1) + \int_{r\in(0,\Delta_1)} lost(z(r), t+r) \cdot cr(d)(x(t+r+\varepsilon_2), t+r+\varepsilon_2).$

M_7 will spontaneously retransmit lost data (very unlikely in practice). Note that this protocol can lead to collisions of different cr actions.

5.3 A FAULTY QUEUE FOR ASYNCHRONOUS MESSAGE TRANSFER.

In this section we will start out from the following description of a queue, that transports data in D from x to y in time Δ.

$$Q = \int_{t>0} \sum_{d\in D} r(d)(x,t) \cdot (s(d)(y, t+\Delta) \parallel Q).$$

This queue is the classical real space absolute time version of example 10.1 of [BAB91a], with port names 1,2 replaced by locations x,y.

We intend to describe a related queue that shows faults. In order to do this, the mode transfer operators $\rightarrow$ and $\hookrightarrow$ of [BE89] are casted in a real time and space setting. Informally, $X \rightarrow Y$ is a process that behaves like X but can at any time start behaving like Y (provided it decides to do so before X has terminated). $X \hookrightarrow Y$ is like $X \rightarrow Y$ with the additional constraint that its first action must be taken from X.

Using the mode transfer operator (to be discussed in detail below), a faulty version of the queue can be given as follows: let $r \in \mathbb{R}$, $n \in \mathbb{N}$, define

$Y(r, n) = s(\perp)(y, n\cdot r) \cdot Y(r, n+1).$

Y(r, n) will produce erroneous signals at regular intervals. Now define Q' by

$$Q' = Q \rightarrow \sum_{n\in\omega} Y(r,\ n).$$

This process allows the queue to transmit data until it switches to mode Y.

Finally, we describe a queue that can be put back in the original mode by means of an action restart(z, t). Here, z is a location from which the queue is controlled.

$$Q'' = Q \rightarrow \Big(\sum_{n\in\omega} Y(r,\ n) \hookrightarrow \int_{t>0} \mathrm{restart}(z, t) \cdot Q''\Big).$$

5.4 MODE TRANSFER.

The equations for the mode transfer operators are as follows ($\alpha \in AS$).

$\alpha(t) \rightarrow X = \alpha(t) + (X \gg t)$	MTT1
$\alpha(t)\cdot X \rightarrow Y = \alpha(t)\cdot(X \rightarrow Y) + (Y \gg t)$	MTT2
$(X + Y) \rightarrow Z = (X \rightarrow Z) + (Y \rightarrow Z)$	MTT3
$(\int_{t\in T} X) \rightarrow Y = \int_{t\in T}(X \rightarrow Y)$ (t not free in Y)	MTT4
$\alpha(t) \hookrightarrow X = \alpha(t)$	DMTT1
$\alpha(t)\cdot X \hookrightarrow Y = \alpha(t)\cdot(X \rightarrow Y)$	DMTT2
$(X + Y) \hookrightarrow Z = (X \hookrightarrow Z) + (Y \hookrightarrow Z)$	DMTT3
$(\int_{t\in T} X) \hookrightarrow Y = \int_{t\in T}(X \hookrightarrow Y)$ (t not free in Y)	DMTT4

Table 17. Mode transfer.

5.5 REMARK.

The equations for mode transfer in [BE89] are as follows:

$$a \rightarrow X = a + X$$
$$\delta \rightarrow X = \delta$$
$$a\cdot X \rightarrow Y = a\cdot(X \rightarrow Y) + Y$$
$$(X + Y) \rightarrow Z = (X \rightarrow Z) + (Y \rightarrow Z).$$

In particular the equation $\delta \rightarrow X = \delta$ is not obvious. However, if a system has deadlocked, it seems impossible to recover from that deadlock in a context $X \rightarrow Y$. In the real time case, we see a more differentiated picture:

$$\delta(2) \rightarrow a(3)\cdot b(4) = \delta(2)$$

while $\delta(2) \rightarrow a(1)\cdot b(4) = \delta(2) + a(1)\cdot b(4)$.

Likewise $c(2) \rightarrow a(3)\cdot b(4) = c(2)$ and $c(2) \rightarrow a(1)\cdot b(4) = c(2) + a(1)\cdot b(4)$.

5.6 REMARK.

We remark that the signal/observation mechanism of [BAB91d] can also be casted in a real time (and space) setting. In that case, this mechanism will provide yet another way to express asynchronous communication.

6. CONCLUSION.

We conclude that we have introduced a setting in which asynchronous communication can be adequately described in classical real space process algebra. This allows to describe communication between processes moving in space (e.g. communication with a satellite).

We claim that we can also describe such communications in the relativistic real space process algebra of [BAB91b]. In that case, because of the presence of the 0 process, the priority operator is not necessary in the description (all timed deadlocks are equated to 0).

REFERENCES.

[BAB88] J.C.M. BAETEN & J.A. BERGSTRA, *Global renaming operators in concrete process algebra,* Information & Computation 78 (3), 1988, pp. 205-245.

[BAB91a] J.C.M. BAETEN & J.A. BERGSTRA, *Real time process algebra,* Formal Aspects of Computing 3 (2), 1991, pp. 142-188. (Report version appeared as report P8916, Programming Research Group, University of Amsterdam 1989.)

[BAB91b] J.C.M. BAETEN & J.A. BERGSTRA, *Real space process algebra,* in Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 96-110.

[BAB91c] J.C.M. BAETEN & J.A. BERGSTRA, *The state operator in real time process algebra,* report P9104, Programming Research Group, University of Amsterdam 1991. To appear in Proc. REX Workshop Real-Time: Theory in Practice, Mook 1991.

[BAB91d] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with signals and conditions,* report CS-R9103, CWI Amsterdam 1991. To appear in Proc. NATO Summer School, Marktoberdorf 1990 (M. Broy et al., eds.), Springer Verlag.

[BABK86] J.C.M. BAETEN, J.A. BERGSTRA & J.W. KLOP, *Syntax and defining equations for an interrupt mechanism in process algebra,* Fund. Inf. IX (2), 1986, pp. 127-168.

[BAW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in Theor. Comp. Sci. 18, Cambridge University Press 1990.

[BE85] J.A. BERGSTRA, *Put and get, primitives for synchronous unreliable message passing,* report LGPS 3, Dept. of Philosophy, Utrecht University 1985.

[BE89] J.A. BERGSTRA, *A mode transfer operator in process algebra,* report P8808b, Programming Research Group, University of Amsterdam 1989.

[BE90] J.A. BERGSTRA, *A process creation mechanism in process algebra,* in: Applications of Process Algebra (J.C.M. Baeten, ed.), Cambridge Tracts in TCS 17, Cambridge University Press 1990, pp. 81-88.

[BEK84] J.A. BERGSTRA & J.W. KLOP, *Process algebra for synchronous communication,* Inf. & Control 60, 1984, pp. 109-137.

[BEKT85] J.A. BERGSTRA, J.W. KLOP & J.V. TUCKER, *Process algebra with asynchronous communication mechanisms,* in: Proc. Seminar on Concurrency (S.D. Brookes, A.W. Roscoe & G. Winskel, eds.), Springer LNCS 197, 1985, pp. 76-95.

[J91a] A. JEFFREY, *A linear time process algebra,* report 61, Programming Methodology Group, Chalmers University 1991, in Proc. CAV 91, Aalborg 1991.

[J91b] A. JEFFREY, *Observation spaces and timed processes,* in Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 332-345.

[J91c] A. JEFFREY, *Translating timed process algebra into prioritized process algebra,* Programming Methodology Group, Chalmers University 1991.

[K91] A.S. KLUSENER, *Completeness in real time process algebra,* report CS-R9106, CWI Amsterdam 1991. Extended abstract in Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 376-392.

[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.

[MU91] D.V.J. MURPHY, *Testing, betting and timed true concurrency,* in Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 439-454.

[RR88] G.M. REED & A.W. ROSCOE, *A timed model for communicating sequential processes,* TCS 58, 1988, pp. 249-261.