ACP WITH SIGNALS

J.A. Bergstra
Programming Research Group, University of Amsterdam
Department of Philosophy, State University of Utrecht

ABSTRACT. New operators are introduced on top of ACP [BK 84] in order to incorporate stable signals in process algebra. Semantically this involves assigning labels to nodes of process graphs. The labels of nodes are called signals. In combination with the operators of BPA, a signal insertion operator allows to describe each finite tree labeled with actions and signals, provided the signals do not occur at leaves of the tree. In a merge processes can observe the signals of concurrent processes. This research was sponsored in part by ESPRIT under contract 432, METEOR.

1 INTRODUCTION

This paper is concerned exclusively with concrete process algebra in the sense of [BB 87]. Concrete process algebra is that part of process algebra that does not involve Milner's silent action τ [Mi 80] or the empty action ϵ due to Vrancken [Vr 86]. The advantage of concrete process algebra is that it admits a fairly clear operational intuition that can serve as a basis for finding algebraic specifications of new operators.

The new feature that this paper contributes to process algebra is the presence of explicit signals of a persistent nature. The original setup of process algebra inherited from Milner's CCS views process semantically as trees of actions. These actions are best thought of as atomic actions because otherwise the intuition behind the axioms becomes rather obscure. Now it was not claimed that the philosophical concept of a process has been analysed in full depth with the introduction of the concepts of process algebra. Some salient features remain unanalysed. For instance real time behavior and true concurrency. A mechanism of particular importance is the presence of visible aspects of the state of a process. Usually in process algebra the state of a process can only be understood (or observed) via the actions that can be performed from that state. In the setup that will be presented here some aspects of the system state are visible not so much through the actions that will follow but much more directly as signals that persist in time for some extended duration. A signal insertion operator will allow to put signals at the root of a process. Using the other operators of process algebra it is then possible to describe processes that have signals on intermediate nodes as well. The main technical complication lies in the signal observation mechanism. Probably I have not defined that mechanism in the full generality that it would admit. In full generality the mechanism should be so powerful as to admit the description of observing continously changing signals in real time.

The reader is supposed to have some familiarity with the axioms systems BPA, PA and ACP for process algebra. In comparison with the extended version [B 88] this discussion is significantly simplified by the absence of terminal signal insertion. Some remarks on the meaning of the operators of ACP can be found in the discussion at the and of this paper. In order to make the paper self contained all axioms of process algebra (ACP) from [BK 84] are repeated.

2 ADDING SIGNALS TO BASIC PROCESS ALGEBRA

The reader is supposed to have some familiarity wih the axioms of BPA. See the discusion for some comments on the meaning of the operators. The core system BPA_δ (basic process algebra with δ) has the following axioms.

A1 $x + y = y + x$
A2 $(x + y) + z = x + (y + z)$
A3 $x + x = x$
A4 $(x + y) \cdot z = x \cdot z + y \cdot z$
A5 $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
A6 $x + \delta = x$
A7 $\delta \cdot x = \delta$

Four new operators and many new axioms will be presented. The first step is to add a finite new sort S of signals equipped with the binary operator & that describes signal combination and with a constant $\emptyset$ denoting the empty signal. Besides the combination operator there is a need for a filtering or intersection operator $\sqcap$. ATS is the small collection of axioms for signal algebra. It explains that & has the algebraic properties of set union and that $\sqcap$ corresponds to set intersection. Indeed the standard interpretation for signal algebra involves a finite set ATS of atomic signals and takes S to be the power set of ATS, identifying atomic signals with the corresponding singleton sets. The axioms of ATS are as follows:

ATS1 $u \,\&\, v = v \,\&\, u$ ATS2 $(u \,\&\, v) \,\&\, w = u \,\&\, (v \,\&\, w)$
ATS3 $u \,\&\, u = u$ ATS4 $u \,\&\, \emptyset = u$
ATS5 $u \sqcap v = v \sqcap u$ ATS6 $u \sqcap (v \sqcap w) = (u \sqcap v) \sqcap w$
ATS7 $u \sqcap u = u$ ATS8 $u \sqcap \emptyset = \emptyset$
ATS9 $u \,\&\, (v \sqcap w) = (u \,\&\, v) \sqcap (u \,\&\, w)$
ATS10 $u \sqcap (v \,\&\, w) = (u \sqcap v) \,\&\, (u \sqcap w)$

The principal new operator to be introduced here is the root signal insertion operator denoted with $[.,.]$. The intuition behind this operator is that it assigns labels (signals) to the states of processes. Root signal insertion places a signal at the root node of a process.

Whereas in process algebra one usually confines oneself to labeling the transitions and perhaps to some labeling of the nodes that is directly related to the mechanism of transition labeling, here it is intended to have labelings of states of processes withe the same status as the labelings of the state transitions by means of atomic actions. With some effort it turns out that an algebraic specification of the resulting notion of processes can be given that indeed constitutes a conservative enrichment of ACP (at least regarding identities between finite closed process expressions). The following equations are added to BPA thus obtaining BPAS (BPA with signals).

RS1 $[u, x] \cdot y = [u, x \cdot y]$
RS2 $[u, x] + y = [u, x + y]$
RS3 $[u, [v, x]] = [u \,\&\, v, x]$
RS4 $[\emptyset, x] = x$

The first axioms expresses the fact that the root of a sequential product is the root of its first component. Axiom RS2 can be given in a more symmetric form as follows:
$[u, x] + [v, y] = [u \,\&\, v, x + y]$. This equation depends on the fact that the roots of two processes in an alternative composition are identified. Therefore signals must be combined. The third axiom expresses the fact that there is no sequential order in the presentation of signals. Of course one might image that a sequential ordering on signals is introduced, but preliminary investigations of this matter have shown that the introduction of sequential ordering is far from obvious. The combination of the signals is taking 'both' of them whereas $x + y$ has to choose between x and y.

An interesting identity that follows with the presence of δ is : $[u,x] = [u,\delta] + x$. This equation is indeed very useful for writing efficient process specifications mainly because it allows to a large extent to work with process algebra expressions that are not cluttered with signal insertions.

EXAMPLES. (i) A traffic light that changes color from green via yellow to red and back to green. As names for the traffic lights one may simply use the natural numbers. For the traffic light with number n, the signals are then $green(n)$, $yellow(n)$ and $red(n)$, the only action is $change(n)$.

$$TL(n) = [green(n), change(n)] \cdot [yellow(n), change(n)] \cdot [red(n), change(n)] \cdot TL(n)$$

(ii) An electric lamp with power switch. In the equations the names for signals and actions speak for themselves. I will assume that the lamp is indexed by a name n from $NAMES = \{hall1, hall2, hall3, kitchen1, kitchen2, living1, living2, living3, staircase, garage, cellar\}$. The lamp is parametrised by two additional parameters: $L(n, x, y)$, $x \in SWITCH = \{on, off\}$, $y \in STATUS = \{defect, functioning\}$. The function sw permutes the elements of $SWITCH$ and the actions $switch(n)$ correspond to switching lamp n. For each $n \in NAMES$ there are two signals: $light(n)$ and $dark(n)$. Together these signals constitute the atomic signals and the signals in the sense of signal algebra are just the finite subsets of this collection. It must be noticed however that the combined signal $light(n) \,\&\, dark(n)$ will not occur in any process in the intended interpretation. The function $F: NAMES \times SWITCH \times STATUS \to SIGNALS$ is defined as follows:

$F(n,x,y) =$ if $x = on$ and $y =$ correct then $light(n)$ else $dark(n)$ fi

Recursion equations for the behavior of the lamp with name n are then as follows.

$$\begin{aligned} L(n, x, functioning) = {} & [F(n, x, functioning), \delta] + \\ & switch(n) \cdot L(n,sw(x), functioning) + \\ & defect(n) \cdot L(n, x, defect) + \\ & get\_new\_lamp(n) \cdot L(n, x, functioning) \end{aligned}$$

```
L(n, x, defect) = [F(n, x, defect), δ] +
                  switch(n) · L(n,sw(x), defect) +
                  get_new_lamp(n) · L(n, x, correct)
```

3 SIGNALS AND MERGE

We can extend the axioms to PAS$^-$, including the free merge (without communication), the left merge with the usual axioms and adding axioms that handle the interaction between the signal insertion operators and the left merge. The superscript $^-$ indicates that there is no feature present that allows the observation of signals, addition of that feature is the subject of sction 4. (Of course the introduction to ACP will require a modification of the merge expansion axiom by adding an additional term for the communication merge, this will be described in section 5.)

M1 $x \parallel y = x \between y + y \between x$

M2 $a \between x = a \cdot x$

M3 $(a \cdot x) \between y = a \cdot (x \parallel y)$

M4 $(x + y) \between z = x \between z + y \between z$

MSI5 $[u, x] \between y = [u, x \between y]$

EXAMPLE. This example refers to the previous example that introduced a collection of lamps in a private house. The simultaneous behavior of the collection of lamps in the living is appropriately described by L(living) = L(living1) || L(living2) || L(living(3)
For the kitchen one obtains: L(kitchen) = L(kitchen1) || L(kitchen2).
For the hall we have L(hall) = L(hall1) || L(hall2)
Composing these one obtains: L(living/kitchen/hall) =
L(kitchen) || L(living) || L(hall).

SIGNALS AND SYNCHRONOUS COMMUNICATION

The axioms system ACPS$^-$ describes the addition of signals to processes with synchronous communication as modeled by ACP. The superscript $^-$ indicates that there is no communication by means of observation of signals. Addition of that feature will involve the addition of two more summands to the merge expansion axiom. The remaining axioms of ACPS$^-$ can now be added without leading to any inconsistency, but it is necessary to add some equations that describe the interaction of left merge and communication merge and the node labels. The axioms CM1-4 replace the axioms M1-4.

C1 $a \mid b = b \mid a$

C2 $(a \mid b) \mid c = a \mid (b \mid c)$

C3 $a \mid \delta = \delta$

CM1 $x \parallel y = x \between y + y \between x + x \mid y$

CM2 $a \between x = a \cdot x$

CM3 $(a \cdot x) \between y = a \cdot (x \parallel y)$

CM4 $(x + y) \between z = (x \between z) + (y \between z)$

MSI1 $[u,x] \between y = [u, x \between y]$

CM5 $a \mid (b \cdot x) = (a \mid b) \cdot x$
CM6 $(a \cdot x) \mid b = (a \mid b) \cdot x$
CM7 $(a \cdot x) \mid (b \cdot y) = (a \mid b) \cdot (x \parallel y)$
CM8 $(x + y) \mid z = (x \mid z) + (y \mid z)$
CM9 $x \mid (y + z) = (x \mid y) + (x \mid z)$

MSI3 $[u, x] \mid y = [u, x \mid y]$
MSI4 $x \mid [u, y] = [u, x \mid y]$

D1 $\partial_H(a) = a$ if $a \notin H$
D2 $\partial_H(a) = \delta$ if $a \in H$
D3 $\partial_H(x + y) = \partial_H(x) + \partial_H(y)$
D4 $\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$
DSI1 $\partial_H([u,x]) = [u, \partial_H(x)]$

4 PROCESSES WITH MERGE AND SIGNAL OBSERVATION

In this section an observation mechanism is added to the features available in process algebra. In order to simplify the discussion, the communication mechanism is first left out in order to be added in section 5. It is assumed that some actions a are able to read the signals of processes that are put in parallel with the process executing a. To this end a subset OBS(A) of the atomic actions is introduced. This set contains the actions that allow non-trivial observations. The reduction of PAS as a special case of PAS$^-$ is found by taking OBS(A) empty. The deadlock constant is never in OBS(A). Moreover a signal observation function is introduced which takes two arguments, an action and a signal and it will return a finite sum of actions. Thus with A the collection of atomic actions including OBS(A), FS(A) is the collection of finite sums of actions in A. Now signal observation is a mapping of type A x SIG $\rightarrow$ FS(A).

I will assume that the boolean algebra of signals is prime. So every signal is the composition of finitely many indecomposable signals. The inspection of a signal will check whether or not some atomic signal is contained in it. Plausible axioms for the signal observation function (./.) are the following:

O1 $a / \emptyset = a$ if $a \in OBS(A)$
O2 $a / u = \delta$ if $a \notin OBS(A)$
O3 $a / (u \,\&\, v) = (a / u) + (a / v)$
O4 $(a / u) / v = a / (u \,\&\, v)$

Notice that axiom O1 is necessary in view of axiom O4, which in turn plays an essential role to ensure associativity of the merge. Notice that axiom O3 is incompatible with the option to have one single action a that observes several signals simultaneously. I have incorporated O3 for two reasons. It simplifies matters, and I did not find any natural examples that need the synchronous

observation of multiple signals. Multiple observations are possible if one leaves out this axiom. Equipped with the signal observation function on atomic actions and signals it is possible to define the result of the observation function on processes with non-trivial (root) signals. The intuitive meaning of x / u is a process that behaves like x be it that the first action of x is viewed as an observation on the signal u.

06 $(x + y) / u = (x / u) + (y / u)$

07 $(x \cdot y) / u = (x / u) \cdot y$

08 $[u, x] / v = [u, x / v]$

09 $\langle x, u \rangle / v = \langle x / v, u \rangle$

THE ROOT SIGNAL OPERATOR

It is useful to extend the system with an operator S which determines the root signal of a process. If $S(x)$ is $\emptyset$ we say that x has a trivial root signal.

S1 $S(a) = \emptyset$

S2 $S(x + y) = S(x) \,\&\, S(y)$

S3 $S(x \cdot y) = S(x)$

S4 $S([u, x]) = u \,\&\, S(x)$

PAS: SIGNAL OBSERVATION AND FREE MERGE

Two new summands must be added to the merge expansion equation in order to obtain the recursion equation for a merge operator that takes signal observation into account as well. This completes the description of the axiom system PAS.

OM $x \parallel y = x \between y + y \between x + (x / S(y)) \between y + (y / S(x)) \between x$

In order to support the intuition for the observation mechanism I will now describe a model for the algebra of signals and the effect of the observation function connected to it.

The model takes a finite set Z of atomic signals as point of departure, z and z' range over Z. $SIG(Z)$, the set of signals over Z is the power-set of Z. The observation actions all have the form $test(z)$ with z an element of Z. These actions represent the intention to observe a signal z. There is a special observation action yes which is the result (confirmation) of a succesful observation. The signal observation function then works as follows.

$test(z) / V =$ if $z \in V$ then yes else $test(z)$ fi.

$yes / V = yes$

One easily verifies the validity of all axioms in this case, for instance

$(test(z) / \{z, z'\}) / \{z\} = yes / \{z\} = yes$ and

$test(z) / (\{z, z'\} \,\&\, \{z\}) = test(z) / \{z, z'\} = yes$.

In connection with this observation function one will use a form of encapsulation which shields off all observations except yes. In this way seen from outside the encapsulation only succesful observations are present.

5 ACPS: SYNCHRONOUS COMMUNICATION AND OBSERVATION

Now I will combine the above specifications of ACPS$^-$ and PAS to obtain ACPS (ACP with signals and signal observation) by adding the two expansion axioms for the merge. The purpose of the following equations is to ensure that all operators except $+$, $\cdot$, $[.,.]$ and $\langle .,. \rangle$ can be eliminated from finite process expressions by means of left to right term rewriting. Notice that the original case of ACP can be viewed similarly with the understanding that all operators except $+$ and $\cdot$ can be eliminated in that case.

Besides taking the axioms for ACPS$^-$ and PAS together while taking the sum of their merge expansion axioms, only one axiom for the communication function is needed; this axiom says that observations will not be involved in any non-trivial communications.

O5 $\quad a \mid b = \delta \quad$ if $a \in OBS(A)$

The new expansion axiom for merge is then as follows:

OCM $\quad x \parallel y = x \mathbin{\mathbb{L}} y + y \mathbin{\mathbb{L}} x + x \mid y + (x \,/\, S(y)) \mathbin{\mathbb{L}} y + (y \,/\, S(x)) \mathbin{\mathbb{L}} x$

If the signal observation function yields δ for every atomic action, the sub-expressions $x \,/\, S(y)$ and $y \,/\, S(x)$ vanish (become equal to δ) and the interaction between processes works exactly as in ACP. ACP is therefore the special case of ACPS if the signal observation function vanishes everywhere. Similarly PA is the special case of ACP if the communication function vanishes everywhere. Obviously there is a situation where the communication function vanishes everywhere but the signal observation function may assume non-trivial values. In such cases one may omit the communication merge.

EXAMPLE. In this example I will elaborate once more on the example of the person in a house with various. The point here is that the person may want to inspect whether or not a light that was switched on indeed functions correctly. If not s/he will replace the bulb by a new one. The actions that must be introduced here are inspections of the signals of the various lamps. Recall that the signals are in this case just finite sets of atomic signals of the form light(n) and dark(n).

```
test(light(n)) / u = yes if light(n) ∈ u
test(dark(n)) / u = yes  if dark(n) ∈ u.
```

The action pswitch(n) can now be replaced, whenever it is assumed to switch the lamp on, by a more involved non-atomic process as follows:

```
pswitch(n) · (test(light(n)) +
test(dark(n)) · put_new_lamp(n))
```

Of course the communication function has to be extended. In particular one needs a communication for the new lamp actions, for instance:

```
put_new_lamp(n) | get_new-lamp(n) = new_lamp(n)
```

Here is process that describes a person entering the house and focussed on repairing broken lights

```
person = enter_front_door · pswitch(hall1) ·
    (test(light(hall1)) +
    test(dark(hall1)) · put_new_lamp(hall1)) ·
    enter_living · (pswitch(living1) ‖ pswitch(living2)) ·
```

```
pswitch(living3) ·
  ( (test(light(living1)) +
       test(dark(living1)) · put_new_lamp(living1)) ||
    (test(light(living2)) +
       test(dark(living2)) · put_new_lamp(living2)) ||
    (test(light(living3)) +
       test(dark(living3)) · put_new_lamp(living3)
  ) ·
enter_kitchen · pswitch(kitchen1) ·
(test(light(n)) + test(dark(n)) · put_new_lamp(n)) ·
leave_kitchen · pswitch(hall1) · enter_living
```

Composing this process person with L(living/kitchen/hall) in the system ACPS will indeed produce a process that shows appropriate interaction of the person with its environment. Notice however that there are many different plausible behaviours of the person. There seems to be no universal generic behavior for person even in this very simple case. Obviously other factors that bear no relation to the switching of light influence the behavior of the person. Nevertheless it is a fair hypothesis to assume that the person behaves for some short time as a process. Thus locally (in time) person is a process in the sense of process algebra, but globally a more complex stucturing mechanism is needed. One can imagine, however, that it is possible to formally incorporate in process algebra a small knowledge base which structures the decision taking process of the person.

6 DISCUSSION

Given a finite alphabet of atomic actions A provided with a commutative and associative communication function and a finite model of SA the initial algebras of BPAS, PAS and ACPS are process domains for finite processes with signals. Process equivalence in these models corresponds to bisimulation equivalence where it is understood that a bisimulation may only relate a pair of nodes in two process graphs if these carry the same signals (or are both terminal nodes in their respective graphs). It follows that the equations of ACPS can be read without hesitation as an algebraic specification for which the initial algebra semantics is the primary meaning.

Of course there is nothing new in the explicit description of process states. JSD [Ja 83] is based on processes but state oriented, Wieringa [Wi 88] uses terminology of knowledge representation as well as the state operator of [BB 87] to have a balanced approach to static and dynamic aspects of process descriptions. [Va 86] provides a process algebra semantics of the parallel programming language POOL which shows that ACP is sufficiently powerful to describe the semantics of a substantial imperative programming language. The relevance of signals is not so much that process algebra without signals lacks expressive power but that some systems can be described in a more natural way using signals.

ACPS attempts not primarily to combine processes and data, as it is done for instance in LOTOS ([Br 87]) for the case of CCS or in PSF [MV 88] for the case of ACP. In both cases an

integration between a process oriented formalism and a formalism for algebraic data type specification is found but no new process concepts are developed. The use of the data types is mainly in facilitating the recursive definition of processes.

Without difficulties an operational semantics with action rules can be given for processes with signals, just as this was done in [vG 86] for ACP. I will collect the transition rules below. Before that I will give a short informal description of the meaning of the operators of ACP. Atomic actions are progams that have the effect of instantaneous events. The alternative composition $X + Y$ of processes X and Y is some process that can evolve like X or like Y. Nothing is presupposed about the mechanism involved in making the choice between X and Y. The product $X \cdot Y$ of X and Y is a process that starts with executing X and then proceeds with executing Y. The merge $X \parallel Y$ executes X and Y in parallel with arbitrary interleaving of atomic actions. The left merge $X \lfloor\!\lfloor Y$ is like the merge but now with the constraint that the first action of $X \lfloor\!\lfloor Y$ is chosen from X. The communication merge $X \mid Y$ is like the merge again but this time with the additional constraint that the first action must be a communication between an action of X and an action of Y. Left merge and communication merge are auxiliary operators used to find an initial algebra specification of merge. Communication at the elvel of individual atomic actions is given by the communication function. $a \mid b$ is the atomic action that results if atomic actions a and b are performed simultaneously (and therefore in communication). If both actions have nothing to communicate the result of $\mid$ is δ. δ is deadlock, a process which has no alternative to proceed. Notice that in a context $\delta + X$ alternatives are added and δ cannot represent deadlock. To finish the description of ACPS here are the action rules for an operational model for processes with signals.

$$a \to^a \surd$$

$$X \to^a X' \Rightarrow X + Y \to^a X' \qquad Y \to^a Y' \Rightarrow X + Y \to^a Y'$$

$$X \to^a X' \Rightarrow X \cdot Y \to^a X' \cdot Y$$

$$X \to^a X' \Rightarrow X \parallel Y \to^a X' \parallel Y \qquad Y \to^a Y' \Rightarrow X \parallel Y \to^a X \parallel Y'$$

$$X \to^a X' \Rightarrow [u, X] \to^a X'$$

$$X \to^a \surd \Rightarrow X + Y \to^a \surd \qquad Y \to^a \surd \Rightarrow X + Y \to^a \surd$$

$$X \to^a \surd \Rightarrow X \cdot Y \to^a X$$

$$X \to^a \surd \Rightarrow X \parallel Y \to^a Y \qquad Y \to^a \surd \Rightarrow X \parallel Y \to^a X$$

$$X \to^a X',\ Y \to^b Y',\ a \mid b = c \Rightarrow X \parallel Y \to^c X' \parallel Y'$$

$$X \to^a X',\ Y \to^b \surd,\ a \mid b = c \Rightarrow X \parallel Y \to^c X'$$

$$X \to^a \surd,\ Y \to^b Y',\ a \mid b = c \Rightarrow X \parallel Y \to^c Y'$$

$$X \to^a \surd,\ Y \to^b \surd,\ a \mid b = c \Rightarrow X \parallel Y \to^c \surd$$

$$X \to^a X',\ a \notin H \Rightarrow \partial_H(X) \to^a \partial_H(X') \qquad X \to^a \surd,\ a \notin H \Rightarrow \partial_H(X) \to^a \surd$$

$$\langle s \mid E \rangle \to^a Y \Rightarrow \langle X \mid E \rangle \to^a Y \qquad \langle s \mid E \rangle \to^a \surd \Rightarrow \langle X \mid E \rangle \to^a \surd$$

In the two above rules $\langle X \mid E \rangle$ is a guarded system of recursion equations that contains the equation $X = s$. $\langle s, E \rangle$ denotes a process as follows: s is a process expression with free

variables in X, for these process variables the unique solution of the equations E is substituted. Last but not least there are the operational rules for signal observation.

$$X \xrightarrow{a} X',\ a/S(Y) = b \neq \delta \Rightarrow X \parallel Y \xrightarrow{b} X' \parallel Y$$
$$X \xrightarrow{a} \surd,\ a/S(Y) = b \neq \delta \Rightarrow X \parallel Y \xrightarrow{b} Y$$
$$Y \xrightarrow{a} Y',\ a/S(X) = b \neq \delta \Rightarrow X \parallel Y \xrightarrow{b} X' \parallel Y$$
$$Y \xrightarrow{a} \surd,\ a/S(X) = b \neq \delta \Rightarrow X \parallel Y \xrightarrow{b} X$$

Adress of the author: University of Amsterdam, Faculty of Mathematics and Computer Sciences, Programming Research Group, Kruislaan 409, 1098 SJ Amsterdam

REFERENCES

[BB 87] J.C.M.Baeten & J.A.Bergstra, Global renaming operators in concrete process algebra, Report P8709, University of Amsterdam, Programming Research Group (1987), to appear in Information and Computation

[B 88] J.A.Bergstra, Process algebra for synchronous communication and observation, University of Amsterdam, Programming Research Group, Report P88XX, (1988)

[BK 84] J.A.Bergstra & J.W.Klop, Process algebra for synchronous communication, Information and Control 60 (1/3), (1984) 109-137

[BCC 86] G.Berry, P.Couronne, & G.Gonthier, Synchronous programming of reactive systems: an introduction to ESTEREL, in Proc.first France-Japan Symposium on Artificial Intelligence and Computer Science, Tokyo, North-Holland, (1986)

[Br 87] (Ed. E. Brinksma) Information Processing Systems- Open Systems Interconnection- LOTOS- A formal description technique based on the temporal ordering of observational behavior, ISO/TC 97/SC 21/20-7-1987

[vGl 87] R.van Glabbeek, Bounded nondeterminism and the approximation induction principle in process algebra, in: proc. STACKS 87 (Eds. F.J.Brandenburg, G.Vidal-Naquet & M.Wirsing) LNCS 247 Springer (1987) 336-347

[MV 88] S.Mauw & G.J.Veltink, A process specification formalism, University of Amsterdam, Programming Research Group, Report XXX (1988)

[Mi 80] R.Milner, A Calculus of Communicating Systems, Springer LNCS, (1980)

[Ja 83] M.Jackson, System development, Prentice Hall, (1983)

[Va 86] F.W.Vaandrager, Process algebra semantics for POOL, Report CS-R8629, Centre for Mathematics and Computer Science, Amsterdam (1986)

[Vr 86] J.L.M.Vrancken, The algebra of communicating processes with empty process, Report FVI 86-01, Programming Research Group, University of Amsterdam, (1986)

[W 88] R.J.Wieringa, Jackson system development analysed in process algebra, Report IR-148, Free University, Amsterdam, Department of Mathematics and Computer Science, (1988)