Process algebra with propositional signals

Eindhoven University of Technology

Department of Mathematics and Computing Science

Process Algebra with Propositional Signals

by

J.C.M. Baeten and J.A. Bergstra

94/49

ISSN 0926-4515

editors: prof.dr. J.C.M. Baeten
prof.dr. M. Rem

Computing Science Report 94/49
Eindhoven, November 1994

Process Algebra with Propositional Signals

J.C.M. Baeten
Department of Computer Science, Eindhoven University of Technology,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

J.A. Bergstra
Programming Research Group, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
and
Department of Philosophy, Utrecht University,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

We consider processes that have transitions labeled with atomic actions, and states labeled with formulas over a propositional logic. These state labels are called signals. A process in a parallel composition may proceed conditionally, dependent on the presence of a signal in the process in parallel. This allows a natural treatment of signal observation.

1991 Mathematics Subject Classification: 68Q60, 68Q10, 68Q40.
1991 CR Categories: F.1.2, D.3.1, F.3.1, D.1.3.
Key words & Phrases: process algebra, process logic, signals.
Note: This research was supported in part by ESPRIT basic research action 7166, CONCUR2. The second author is also partially supported by ESPRIT basic research action 6454, CONFER.

1. INTRODUCTION.

This paper can be viewed as a revision and simplification of [BAB92] in which we have introduced so-called signals as labels for states in processes (see also [BRO90]). Though useful in a multitude of examples, it has turned out that the mechanism of signal observation of [BAB92] is quite complex. The approach taken was that actions observe signals. What we propose here is to require that the signals are propositions (i.e. elements of a boolean algebra; this is consistent with [BAB92]) and then to use tests to read off information from these signals. In this way, conditions in propositional expressions (written as $\phi :\rightarrow x$, or $x \lhd \phi \rhd y$) and propositional signals are complementary. A mechanism to localise or hide the propositional signals is important. In summary, our development is based on the position:

- the visible part (signal) of the state of a process is a proposition.

Whatever the merits of this position, what we do establish is that it is a consistent position, and that it allows a wide range of examples.

The introduction of propositional signals in the context of process algebra occurs to us as a necessary step, it completes the picture that emerges if conditional process expressions are introduced. Indeed, consider an expression $x \lhd \phi \rhd y$. If ϕ is true or false, this is just x or y. But in the more general case that ϕ ranges over a class of propositions, what determines the meaning of $x \lhd \phi \rhd y$? An

answer is: ϕ is to be evaluated over a state. This leads one into state operators (as in [BAB88]) or global states (see [GRP94]), thus departing from the core of process algebra where every dynamic entity is a process.

So we feel that the primary motivation of this paper is a conceptual one and that additional motivation in terms of potential applications is both premature and superfluous. This is not meant to imply that we are pessimistic about applications. It rather is the case that we would propose to view process algebra with propositional signals as a subject in pure logic at least initially. Many extensions or modifications can be imagined: first order signals, higher order signals, infinitary and non-classical logics for the entailment relation between signals and conditions, modal and temporal logics for processes with propositional signals.

We are not aware of any previous work aiming at objectives similar to our present ones. The present approach is also followed in [BEP94]. Clearly, our approach, based on ACP [BEK84] can be adapted to CCS [MIL80], MEIJE [AUB84] or ATP [NIS94] without much effort. Adaptation to CSP [BRHR84] is more involved dure to the different models, based on failure or ready sets.

2. BASIC PROCESS ALGEBRA WITH PROPOSITIONAL SIGNALS.

2.1 BPA WITH INACTION AND NONEXISTENCE.

Let A be a finite set. The elements of A will be called atomic actions. Every atomic action is an element of P, the sort of processes. There are also two binary operators on P, viz. $+$ (alternative composition) and $\cdot$ (sequential composition). The core system BPA (Basic Process Algebra) over this signature has the axioms A1-5 of table 1 below ($x,y,z \in P$). The constant δ denoting inaction (or deadlock) is added to the language with axioms A6,7. In this paper, we introduce a new constant for process algebra, viz. $\bot$. This constant stands for nonexistence, we will need it when we introduce signals further on. It is axiomatized by axioms NE1-3 of table 1 ($x \in P, a \in A$). Nonexistence stands for an inconsistent state of a process: such a state can never be exited (NE1,2) and also, it is impossible to enter such a state from a consistent state (NE3). This signature and these axioms together constitute the theory $BPA_\bot$, Basic Process Algebra with inaction and nonexistence.

$x + y = y + x$	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
$x + \delta = x$	A6
$\delta\cdot x = \delta$	A7
$x + \bot = \bot$	NE1
$\bot\cdot x = \bot$	NE2
$a\cdot\bot = \delta$	NE3

TABLE 1. $BPA_\bot$.

2.2 CONDITIONALS.

Besides the sort of processes P, we will have a second sort B. Elements of this sort are propositional logic formulas over a set of basic propositional variables $P_1, ..., P_n$ with constants T, F (true, false) and operators $\vee,\wedge,\supset,\neg$ (disjunction, conjunction, implication, negation). In derivations we can use identities of propositional logic. We use letters ϕ,ψ to range over B.

As in [BAB92], we use the *guarded command*. $.:\!\rightarrow\!.: B \times P \rightarrow P$. The expression $\phi:\!\rightarrow x$ is read as **if** ϕ **then** x. We have the basic axioms of table 2 below, using the numbering of [BAB92].

$T :\!\rightarrow x = x$	GC1
$F :\!\rightarrow x = \delta$	GC2
$\phi :\!\rightarrow \delta = \delta$	GC9
$\phi :\!\rightarrow (x + y) = (\phi :\!\rightarrow x) + (\phi :\!\rightarrow y)$	GC10
$\phi :\!\rightarrow (x\cdot y) = (\phi :\!\rightarrow x)\cdot y$	GC13
$(\phi \vee \psi) :\!\rightarrow x = (\phi :\!\rightarrow x) + (\psi :\!\rightarrow x)$	GC11
$\phi :\!\rightarrow (\psi :\!\rightarrow x) = (\phi \wedge \psi) :\!\rightarrow x$	GC12

TABLE 2. Conditionals over propositional logic.

2.3 ROOT AND TERMINAL SIGNAL EMISSION OPERATORS.

The next operators to be introduced are the signal emission operators. $\nearrow$ is the root signal emission operator and $\nwarrow$ is the terminal signal emission operator. The intuition behind these operators is that both assign labels (signals) to the states of processes. Root signal emission places a signal at the root node of a process. Terminal signal emission places one and the same signal at each terminal node of a process. If one is interested solely in processes that emit signals exclusively in nonterminal states one may as well forget about the terminal signal emission operator. Leaving out all axioms involving terminal emission from the coming sections one will obtain an appropriate description of root signal

emission. The following equations are added to $BPA_\perp$ thus obtaining BPAps (BPA with Propositional Signals).

$(\phi \nwarrow x)\cdot y = \phi \nwarrow (x\cdot y)$	RSE1
$(\phi \nwarrow x) + y = \phi \nwarrow (x + y)$	RSE2
$\phi \nwarrow (\psi \nwarrow x) = (\phi \wedge \psi) \nwarrow x$	RSE3
$T \nwarrow x = x$	RSE4
$F \nwarrow x = \perp$	RSE5
$\phi \nwarrow \perp = \perp$	RSE6
$\phi :\rightarrow (\psi \nwarrow x) = (\phi \supset \psi) \nwarrow (\phi :\rightarrow x)$	RSE7
$\phi \nwarrow (\phi :\rightarrow x) = \phi \nwarrow x$	RSE8

TABLE 3. Root signal emission.

The first axiom 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:

$$(\phi \nwarrow x) + (\psi \nwarrow y) = (\phi \wedge \psi) \nwarrow (x + y).$$

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 imagine that a sequential ordering on signals is introduced, but we think that the introduction of such a sequential ordering is far from obvious (it also leads to problems concerning the associativity of the parallel composition operator). The combination of the signals is taking 'both' of them whereas $x + y$ has to choose between x and y. As an example, consider the following derivation:

$$a\cdot((\phi \nwarrow b) + (\neg\phi \nwarrow b)) = a\cdot((\phi \wedge \neg\phi) \nwarrow b) = a\cdot(F \nwarrow b) = a\cdot\perp = \delta.$$

The last axiom RSE8 is the *signal inspection rule*. If a signal ϕ is emitted, then ϕ holds in the current state (this is why the signal F denotes an inconsistent state). Note the following generalisation of RSE8:

$$\phi \nwarrow (\psi :\rightarrow x) = \phi \nwarrow (\phi :\rightarrow (\psi :\rightarrow x)) = \phi \nwarrow ((\phi \wedge \psi) :\rightarrow x).$$

The equations below regard terminal signal emission.

An interesting identity that follows is the following:

$$\phi \nwarrow x = (\phi \nwarrow \delta) + x.$$

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 emissions.

The axiom system BPAps, Basic Process Algebra with Propositional Signals, consists of all axioms from tables 1-4.

$(x \cdot y) \leftarrow \phi = x \cdot (y \leftarrow \phi)$	TSE1
$(x + y) \leftarrow \phi = x \leftarrow \phi + y \leftarrow \phi$	TSE2
$(x \leftarrow \phi) \leftarrow \psi = x \leftarrow (\phi \wedge \psi)$	TSE3
$x \leftarrow T = x$	TSE4
$a \leftarrow F = \delta$	TSE5
$\delta \leftarrow \phi = \delta$	TSE6
$\bot \leftarrow \phi = \bot$	TSE7
$(\phi :\rightarrow x) \leftarrow \psi = \phi :\rightarrow (x \leftarrow \psi)$	TSE8
$(x \leftarrow \phi) \cdot y = x \cdot (\phi \leftarrow y)$	TRSE1
$(\phi \leftarrow x) \leftarrow \psi = \phi \leftarrow (x \leftarrow \psi)$	TRSE2

TABLE 4. Remaining axioms of BPAps.

2.4 BASIC TERMS.

Define a set of basic terms $\mathcal{B}$ as follows.

i. $\bot \in \mathcal{B}$

ii. $\phi \in B - \{F\} \Rightarrow \phi \leftarrow \delta \in \mathcal{B}$

iii. $\phi, \psi \in B - \{F\}, a \in A \Rightarrow (\phi :\rightarrow a \leftarrow \psi) \in \mathcal{B}$

iv. $\phi \in B - \{F\}, a \in A, t \in \mathcal{B} \Rightarrow (\phi :\rightarrow a \cdot t) \in \mathcal{B}$

v. $t, s \in \mathcal{B} \Rightarrow t + s \in \mathcal{B}$

Note that each basic term can be written as $\bot$ or in the form:

$$\zeta \leftarrow \delta + \sum_{i=1}^{n} \phi_i :\rightarrow a_i \cdot x_i + \sum_{j=1}^{m} \psi_j :\rightarrow b_j \leftarrow \chi_j \qquad (\zeta, \phi_i, \psi_j, \xi_j \in B - \{F\}, a_i \in A, b_j \in A_\delta, x_i \in \mathcal{B}),$$

or, equivalently,

$$\zeta \leftarrow (\sum_{i=1}^{n} \phi_i :\rightarrow a_i \cdot x_i + \sum_{j=1}^{m} \psi_j :\rightarrow b_j \leftarrow \chi_j).$$

When a basic term has this form, we call ζ its *root signal*, and the subterms $\phi_i :\rightarrow a_i \cdot x_i$, $\psi_j :\rightarrow b_j \leftarrow \chi_j$ its *summands*.

2.5 BASIC TERM LEMMA. For all closed terms s there is a basic term t such that BPAps $\vdash t = s$.

SKETCH OF PROOF. Basically, this follows from the fact that the term rewriting system consisting of axioms A4,5 from table 1 together with all axioms from tables 2,3,4 is strongly normalising. This can be proved by using the method of the lexicographical path ordering (making the signature one-sorted, adding rewrite rules for propositional logic, taking the ordering $\leftarrow > :\rightarrow > \cdot > + > \delta$, $+ > \leftarrow >$

$\bot,\wedge,\supset$, giving $\cdot$ the lexicographical status for the first argument, and $:\!\rightarrow$, $\leftarrow\!\!\wedge$, $\wedge\!\!\rightarrow$ for the second argument). Each normal form of this term rewriting system can easily be converted into a basic term.

2.6 STRUCTURED OPERATIONAL SEMANTICS.

We proceed to give the semantics of BPAps using structured operational rules (SOS).

The semantics uses the following predicates and relations on closed terms:

- $x \xrightarrow{\phi,a} x'$ term x can do an a-step under condition ϕ to term x'
- $x \xrightarrow{\phi,a} \psi$ term x can do a terminating a-step under condition ϕ leaving terminal signal ψ
- $s_\rho(x) = \phi$ the root signal of x is ϕ.

Plotkin-style rules for the step relations and step predicates are given in table 5, the rules for the root signal predicate are given in the form of axioms in table 6. This SOS specification is in the *path* format of [BAV93].

$$a \xrightarrow{T,a} T$$

$$\frac{x \xrightarrow{\phi,a} x',\ s_\rho(x+y) \neq F}{x+y \xrightarrow{\phi,a} x',\ y+x \xrightarrow{\phi,a} x'} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ s_\rho(x+y) \neq F}{x+y \xrightarrow{\phi,a} \psi,\ y+x \xrightarrow{\phi,a} \psi}$$

$$\frac{x \xrightarrow{\phi,a} x'}{x\cdot y \xrightarrow{\phi,a} x'\cdot y} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ \psi\wedge s_\rho(y) \neq F}{x\cdot y \xrightarrow{\phi,a} \psi \leftarrow\!\!\wedge y}$$

$$\frac{x \xrightarrow{\phi,a} x',\ \psi\wedge s_\rho(x) \neq F}{\psi \wedge\!\!\rightarrow x \xrightarrow{\phi,a} x'} \qquad \frac{x \xrightarrow{\phi,a} \chi,\ \psi\wedge s_\rho(x) \neq F}{\psi \wedge\!\!\rightarrow x \xrightarrow{\phi,a} \chi}$$

$$\frac{x \xrightarrow{\phi,a} x'}{x \leftarrow\!\!\wedge \psi \xrightarrow{\phi,a} x \leftarrow\!\!\wedge \psi} \qquad \frac{x \xrightarrow{\phi,a} \chi,\ \chi\wedge\psi \neq F}{x \leftarrow\!\!\wedge \psi \xrightarrow{\phi,a} \chi\wedge\psi}$$

$$\frac{x \xrightarrow{\phi,a} x'}{(\psi :\!\rightarrow x) \xrightarrow{\phi\wedge\psi,a} x'} \qquad \frac{x \xrightarrow{\phi,a} \chi}{(\psi :\!\rightarrow x) \xrightarrow{\phi\wedge\psi,a} \chi}$$

TABLE 5. SOS rules.

$s_\rho(\bot) = F$	RSO0
$s_\rho(a) = T$	RSO1
$s_\rho(x + y) = s_\rho(x) \wedge s_\rho(y)$	RSO2
$s_\rho(x \cdot y) = s_\rho(x)$	RSO3
$s_\rho(\phi \leftarrow x) = \phi \wedge s_\rho(x)$	RSO4
$s_\rho(x \rightarrow \phi) = s_\rho(x)$	RSO5
$s_\rho(\phi :\rightarrow x) = \phi \supset s_\rho(x)$	RSO6

TABLE 6. Root signal operator.

Based on this operational semantics involving conditions on the arrows comes a new definition for bisimulation. Instead of just requiring matching actions, we also require matching conditions; however, one transition on one side may have to be matched with several transitions on the other side, depending on the truth value of the propositional constants. Therefore, the following defintion starts from the set of valuations of the propositional constants, i.e. all mappings $v: \{P_1,...,P_n\} \rightarrow BOOL$. Each such mapping naturally extends to a mapping on all formulas. We write $\phi=\psi$ (also in the rules above) iff for all valuations v, $v(\phi) = T$ iff $v(\psi) = T$. Similarly, $\phi\neq\psi$ iff there is a valuation v with $v(\phi)=T$ and $v(\psi)=F$, or $v(\phi)=F$ and $v(\psi)=T$.

Then we say that a relation R on closed terms is a *(strong) bisimulation* when the following holds:

i. if xRy then $s_\rho(x) = s_\rho(y)$

ii. if xRy and $x \xrightarrow{\phi,a} x'$, then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there is a condition ψ and an expression y' such that $v(\psi) = T$, $y \xrightarrow{\psi,a} y'$ and $x'Ry'$

iii. if xRy and $y \xrightarrow{\phi,a} y'$, then for all valuations v such that $v(s_\rho(y) \wedge \phi) = T$, there is a condition ψ and an expression y' such that $v(\psi) = T$, $x \xrightarrow{\psi,a} x'$ and $x'Ry'$

iv. if xRy and $x \xrightarrow{\phi,a} \psi$ then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are conditions ϕ',ψ' with $v(\phi') = T$, $\psi = \psi'$ and $y \xrightarrow{\phi',a} \psi'$

v. if xRy and $y \xrightarrow{\phi,a} \psi$ then for all valuations v such that $v(s_\rho(y) \wedge \phi) = T$, there are conditions ϕ',ψ' with $v(\phi') = T$, $\psi = \psi'$ and $x \xrightarrow{\phi',a} \psi'$.

We call two expressions x,y (strongly) bisimilar, notated $x \leftrightarroweq y$, if there is a (strong) bisimulation relating x and y. We state the following proposition without proof.

2.7 PROPOSITION. Bisimulation is an congruence relation on process expressions.

As a consequence, we can consider the algebraic structure $\mathbb{P}/\leftrightarroweq$ of process expressions modulo bisimulation equivalence.

2.8 THEOREM (SOUNDNESS). $\mathbb{P}/\leftrightarroweq \models$ BPAps.

PROOF: By the previous proposition, it is enough to verify each axiom separately. We confine ourselves to give the bisimulation relation. Note that $\mathbb{P}/\leftrightarroweq \not\models \bot = \delta$, since $s_\rho(\bot) = F \neq T = s_\rho(\delta)$.

For axiom A1, take the relation relating left-hand and right-hand side and relating each term to itself. A2,3,4 go similarly. For A5, relate in addition all pairs of the form $x\cdot(y\cdot z)$ to $(x\cdot y)\cdot z$, and all pairs of the form $(\phi \leftharpoonup x)\cdot y$ to $\phi \leftharpoonup (x\cdot y)$. A6 goes like A1, and for A7 it suffices to relate right-hand and left-hand side. NE1,2,3 go like A7.

GC1,10,13,12 go like A1, GC2,9 like A7. GC11 also goes like A1, but note that here we use the fact that for a valuation v, $v((\phi\wedge\chi) \vee (\psi\wedge\chi)) = T$ iff $v(\phi\wedge\chi) = T \vee v(\psi\wedge\chi) = T$.

RSE1,2,3,4,7 go like A1, RSE5,6 like A7. RSE8 also goes like A1, but here we also use the fact that we only need to consider valuations that make the root signal true. For TSE1, relate all terms to themselves, and all terms of the form $(x\cdot y) \leftharpoondown \phi$ to $x\cdot(y \leftharpoondown \phi)$. TSE2,8, TRSE2 go like A1. For TSE3, relate all terms to themselves, and all terms of the form $(x \leftharpoondown \phi) \leftharpoondown \psi$ to $x \leftharpoondown (\phi\wedge\psi)$. For TSE4, relate all terms to themselves, and all terms of the form $x \leftharpoondown T$ to x. TSE5,6,7 go like A7. For TRSE1, relate all terms to themselves, all terms of the form $(x \leftharpoondown \phi)\cdot y$ to $x\cdot(\phi \leftharpoonup y)$, and all terms of the form $\phi \leftharpoonup (\psi \leftharpoonup x)$ to $(\phi\wedge\psi) \leftharpoonup x$.

For basic terms, there is a direct relation between syntax and semantics.

2.9 LEMMA. Let $t \in \mathcal{B}$.

i. The root signal of t is $s_\rho(t)$

ii. $t \xrightarrow{\phi,a} s$ iff $\phi :\rightarrow a\cdot s$ is a summand of t

iii. $t \xrightarrow{\phi,a} \psi$ iff $\phi :\rightarrow a \leftharpoondown \psi$ is a summand of t

2.10 THEOREM (COMPLETENESS). Let t,s be two closed BPAps terms. Then $x \leftrightarroweq y$ implies BPAps $\vdash t=s$.

PROOF: By the basic term lemma, it is enough to prove this for basic terms. The proof can be completed using lemma 2.9.

As a corrolary, we have $\mathbb{P}/\leftrightarroweq \models t=s \Leftrightarrow$ BPAps $\vdash t=s$ for all closed t,s.

2.11 GLOBAL SIGNAL EMISSION.

In the next section, we will extend BPAps with parallel composition. There, we will need as an extra operator the global signal emission operator, that adds a signal to each state of a process. We give axioms for this operator in table 7, and semantical rules in table 8.

$\phi \lceil\!\rceil \bot = \bot$	GSE0
$\phi \lceil\!\rceil a = \phi \barwedge a \barwedge \phi$	GSE1
$\phi \lceil\!\rceil (x + y) = (\phi \lceil\!\rceil x) + (\phi \lceil\!\rceil y)$	GSE2
$\phi \lceil\!\rceil (x \cdot y) = (\phi \lceil\!\rceil x) \cdot (\phi \lceil\!\rceil y)$	GSE3
$\phi \lceil\!\rceil (\psi \barwedge x) = \psi \barwedge (\phi \lceil\!\rceil x)$	GSE4
$\phi \lceil\!\rceil (x \barwedge \psi) = (\phi \lceil\!\rceil x) \barwedge \psi$	GSE5
$\phi \lceil\!\rceil (\psi :\!\rightarrow x) = \psi :\!\rightarrow (\phi \lceil\!\rceil x) + \neg\psi :\!\rightarrow (\phi \barwedge \delta)$	GSE6

TABLE 7. Global signal emission.

$$\frac{x \xrightarrow{\phi,a} x',\ \psi \wedge s_\rho(x) \neq F}{\psi \lceil\!\rceil x \xrightarrow{\phi,a} \psi \lceil\!\rceil x'} \qquad \frac{x \xrightarrow{\phi,a} \chi,\ \psi \wedge s_\rho(x) \neq F,\ \chi \wedge \psi \neq F}{\psi \lceil\!\rceil x \xrightarrow{\phi,a} \psi \wedge \chi}$$

$$s_\rho(\phi \lceil\!\rceil x) = \phi \wedge s_\rho(x)$$

TABLE 8. Operational semantics of global signal emission.

With the help of the global signal emission operator, we can define a notion of invariance:
DEFINTION: ϕ is an *invariant* of x if $\phi \barwedge x = \phi \lceil\!\rceil x$.

2.12 ROOT SIGNAL OPERATOR AND ROOT SIGNAL DELETION OPERATOR.

We used the root signal operator s_ρ in the operational semantics. We can also add this operator to the theory with the axioms of table 6. The operator s_ρ determines the root signal of a process. If $s_\rho(x) = T$ we say that x has a *trivial root signal*, otherwise x has a non-trivial root signal. Processes that were studied until now in the context of process algebra always have a trivial root signal. We can also define an operator p_ρ, that removes the root signal from its argument. It remains to be seen if this operator is useful.

Notice that the equation $s_\rho(x \barwedge \phi) = s_\rho(x)$ is derivable:

$$s_\rho(x \barwedge \phi) = s_\rho((x \barwedge \phi) \cdot y) = s_\rho(x \cdot (\phi \barwedge y)) = s_\rho(x).$$

Also $x = s_\rho(x) \barwedge p_\rho(x)$ will now be derivable for finite closed process expressions. As a rewrite rule it is useless, however, because it will immediately introduce an infinite loop.

$p_\rho(\bot) = \bot$	RSD0
$p_\rho(a) = a$	RSD1
$p_\rho(x + y) = s_\rho(x+y){:}\rightarrow(p_\rho(x) + p_\rho(y)) + \neg s_\rho(x+y){:}\rightarrow\bot$	RSD2
$p_\rho(x{\cdot}y) = p_\rho(x){\cdot}y$	RSD3
$p_\rho(\phi \leftwedge x) = (\phi{:}\rightarrow p_\rho(x)) + (\neg\phi{:}\rightarrow\bot)$	RSD4
$p_\rho(x \leftarrow\!\wedge \phi) = p_\rho(x) \leftarrow\!\wedge \phi$	RSD5
$p_\rho(\phi :\rightarrow x) = \phi :\rightarrow p_\rho(x)$	RSD6

TABLE 9. Root signal deletion operator.

2.13 SIGNAL HIDING.

An important operator in applications is the signal hiding operator Δ, that hides a propositional constant P. We give axioms based on the structure of basic terms in table 10, and provide semantics in table 11.

$P \Delta \bot = \bot$	SH0
$P \Delta (\phi \leftwedge \delta) = (\phi[T/P] \vee \phi[F/P]) \leftwedge \delta$	SH1
$P \Delta (x + y) = P \Delta (s_\rho(x+y) \leftwedge x) + P \Delta (s_\rho(x+y) \leftwedge y)$	SH2
$P \Delta (\phi \leftwedge \psi{:}\rightarrow a{\cdot}x) = (\phi[T/P] \vee \phi[F/P]) \leftwedge ((\phi\wedge\psi)[T/P]{:}\rightarrow a{\cdot}(P \Delta x) + (\phi\wedge\psi)[F/P]{:}\rightarrow a{\cdot}(P \Delta x))$	SH3
$P \Delta (\phi \leftwedge \psi{:}\rightarrow a \leftarrow\!\wedge \chi) = (\phi[T/P] \vee \phi[F/P]) \leftwedge ((\phi\wedge\psi)[T/P]{:}\rightarrow a \leftarrow\!\wedge (\chi[T/P] \vee \chi[F/P]) + (\phi\wedge\psi)[F/P]{:}\rightarrow a \leftarrow\!\wedge (\chi[T/P] \vee \chi[F/P]))$	SH4

TABLE 10. Signal hiding.

$$\frac{x \xrightarrow{\phi,a} x',\ \psi \equiv (s_\rho(x)\wedge\phi)[T/P]}{P\Delta x \xrightarrow{\psi,a} P\Delta x'} \qquad \frac{x \xrightarrow{\phi,a} x',\ \psi \equiv (s_\rho(x)\wedge\phi)[F/P]}{P\Delta x \xrightarrow{\psi,a} P\Delta x'}$$

$$\frac{x \xrightarrow{\phi,a} \chi,\ \psi \equiv (s_\rho(x)\wedge\phi)[T/P]}{P\Delta x \xrightarrow{\psi,a} \chi[T/P]\vee\chi[F/P]} \qquad \frac{x \xrightarrow{\phi,a} \chi,\ \psi \equiv (s_\rho(x)\wedge\phi)[F/P]}{P\Delta x \xrightarrow{\psi,a} \chi[T/P]\vee\chi[F/P]}$$

$$s_\rho(P \Delta x) = s_\rho(x)[T/P] \vee s_\rho(x)[F/P]$$

TABLE 11. Operational semantics of signal hiding.

Example: $P \Delta (a\cdot(P \leftarrow b) + a\cdot(\neg P \leftarrow b)) = a\cdot(T \leftarrow (T{:}\rightarrow b)) + a\cdot(T \leftarrow (T{:}\rightarrow b)) = a\cdot b$. If we assume the linear time law $a\cdot(x + y) = a\cdot x + a\cdot y$, this leads to the unwanted identity $a\cdot b = \delta$ (combine with the example in 2.3). Thus, this theory only exists in a branching time setting.

The global signal emission operator of 2.11, the root signal operator and root signal deletion operator of 2.12 and the signal hiding operator here can all be eliminated from closed terms, using the axioms given. Thus, we have the basic term lemma also for this extended signature. It is not difficult to establish that the extended theory is a conservative extension of BPAps, and that the axiomatisation is sound and complete for the term model modulo bisimulation (use the recipe of [BAV94]).

2.14 ITERATION.

We will not deal with full recursion in this paper. It is enough to consider linear recursion and iteration. For iteration, we use the operator * (Binary Kleene Star) of [BEBP94]. We present axioms in table 12, operational semantics in table 13. We have one extra axiom for the Kleene star and terminal signal emission.

$x\cdot(x^*y) + y = x^*y$	BKS1
$x^*(y\cdot z) = (x^*y)\cdot z$	BKS2
$x^*(y\cdot((x + y)^*z) + z) = (x + y)^*z$	BKS3
$x^*(y \leftarrow \phi) = (x^*y) \leftarrow \phi$	BKSTE

TABLE 12. Binary Kleene Star.

$$\frac{x \xrightarrow{\phi,a} x',\ s_\rho(x^*y) \neq F}{x^*y \xrightarrow{\phi,a} x'\cdot(x^*y)} \qquad \frac{y \xrightarrow{\phi,a} y',\ s_\rho(x^*y) \neq F}{x^*y \xrightarrow{\phi,a} y'}$$

$$\frac{x \xrightarrow{\phi,a} \psi,\ s_\rho(x^*y)\wedge\psi \neq F}{x^*y \xrightarrow{\phi,a} \psi \leftarrow (x^*y)} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ s_\rho(x^*y) \neq F}{x^*y \xrightarrow{\phi,a} \psi}$$

$$s_\rho(x^*y) = s_\rho(x) \wedge s_\rho(y)$$

TABLE 13. Operational semantics of binary Kleene star.

2.15 LINEAR RECURSION.

A recursion equation is *linear* if it is of the form:

$$X = \zeta \leftarrow \delta + \sum_{i=1}^{n} \phi_i{:}\rightarrow a_i\cdot X_i + \sum_{j=1}^{m} \psi_j{:}\rightarrow b_j \leftarrow \chi_j$$

where $\zeta,\phi_i,\psi_j,\xi_j \in B - \{F\}$, $a_i \in A$, $b_j \in A_\delta$, and the X, X_i are recursion variables. The semantics for processes given by systems of such equations is easily given: for an equation as above we have for each summand a transition, of one the forms

$$X \xrightarrow{\phi_i,a_i} X_i \qquad\qquad X \xrightarrow{\psi_j,b_j} \chi_j$$

Recursion equations in the following can simply be brought into this form.

3. PARALLEL COMPOSITION.

In this section, we extend the basic theory of section 2 with parallel composition. First, we consider parallel composition without synchronisation or communication, the so-called free merge.

3.1 PAPS.

The theory PAps, Process Algebra with Propositional Signals, extends BPAps with operators $\overset{\rightarrow}{\lceil}$, $\|$, $\lfloor\!\lfloor$, and the axioms of tables 7 and 14.

$x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$	M1
$(a \overset{\phi}{\nwarrow}) \lfloor\!\lfloor x = a \cdot (\phi \overset{\rightarrow}{\lceil} x)$	M2TS
$a \cdot x \lfloor\!\lfloor y = a \cdot (x \| y)$	M3
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	M4
$(\phi \overset{\nwarrow}{\nearrow} x) \lfloor\!\lfloor y = \phi \overset{\nwarrow}{\nearrow} (x \lfloor\!\lfloor y)$	MRS
$(\phi :\!\rightarrow x) \lfloor\!\lfloor y = \phi :\!\rightarrow (x \lfloor\!\lfloor y)$	MGC

TABLE 14. Free merge.

Note: $\bot \lfloor\!\lfloor x = (F \overset{\nwarrow}{\nearrow} a) \lfloor\!\lfloor x = F \overset{\nwarrow}{\nearrow} (a \lfloor\!\lfloor x) = \bot$, and $a \lfloor\!\lfloor x = (a \overset{\nwarrow}{\nearrow} T) \lfloor\!\lfloor x = a \cdot (T \overset{\rightarrow}{\lceil} x)$, where the last expression can be proven equal to $a \cdot x$ for all closed terms.

3.2 SIGNAL INSPECTION.

Now we have all the ingredients necessary to describe the inspection of an emitted signal. A very simple example will serve to make the point. Let us consider a traffic light. The set of propositional constants is {green, yellow, red}.

$TL(green) = (green \wedge \neg yellow \wedge \neg red) \overset{\nwarrow}{\nearrow} change \cdot TL(yellow)$

$TL(yellow) = (\neg green \wedge yellow \wedge \neg red) \overset{\nwarrow}{\nearrow} change \cdot TL(red)$

$TL(red) = (\neg green \wedge \neg yellow \wedge red) \overset{\nwarrow}{\nearrow} change \cdot TL(green)$.

Now we describe a careful car driver.

$CD = approach \cdot ((\neg green :\!\rightarrow stop) \cdot (green :\!\rightarrow start) \cdot drive + (green :\!\rightarrow drive))$.

Expression $TL(x) \| CD$ now describes a correct interaction between light and driver.

3.3 ACPps.

The theory ACPps, Algebra of Communicating Processes with Propositional Signals, extends PAps with operators $\mid$, ∂_H, s_ρ, and replaces the axioms of table 14 by the axioms of the root signal operator and the axioms in table 15 below. We assume given a partial commutative and associative binary function on A, the communication function γ.

$a \mid b = \gamma(a,b)$	if defined	CF1
$a \mid b = \delta$	otherwise	CF2
$x \| y = x \mathbin{\mathbb{L}} y + y \mathbin{\mathbb{L}} x + x \mid y$		CM1
$(a \overset{\phi}{\frown}) \mathbin{\mathbb{L}} x = a \cdot (\phi \rightharpoonup x)$		CM2TS
$a \cdot x \mathbin{\mathbb{L}} y = a \cdot (x \| y)$		CM3
$(x + y) \mathbin{\mathbb{L}} z = x \mathbin{\mathbb{L}} z + y \mathbin{\mathbb{L}} z$		CM4
$(\phi \overset{\cdot}{\frown} x) \mathbin{\mathbb{L}} y = \phi \overset{\cdot}{\frown} (x \mathbin{\mathbb{L}} y)$		CMRS1
$(\phi :\!\rightarrow x) \mathbin{\mathbb{L}} y = \phi :\!\rightarrow (x \mathbin{\mathbb{L}} y)$		CMGC1
$(a \overset{\phi}{\frown}) \mid (b \overset{\psi}{\frown}) = (a \mid b) \overset{\phi \wedge \psi}{\frown}$		CMTS
$(a \overset{\phi}{\frown}) \mid b \cdot x = (a \mid b) \cdot (\phi \rightharpoonup x)$		CM5TS
$a \cdot x \mid (b \overset{\psi}{\frown}) = (a \mid b) \cdot (\phi \rightharpoonup x)$		CM6TS
$a \cdot x \mid b \cdot y = (a \mid b) \cdot (x \| y)$		CM7TS
$(x+y) \mid z = x \mid z + y \mid z$		CM8
$x \mid (y+z) = x \mid y + x \mid z$		CM9
$(\phi \overset{\cdot}{\frown} x) \mid y = \phi \overset{\cdot}{\frown} (x \mid y)$		CMRS2
$x \mid (\psi \overset{\cdot}{\frown} y) = \phi \overset{\cdot}{\frown} (x \mid y)$		CMRS3
$(\phi :\!\rightarrow x) \mid y = \phi :\!\rightarrow (x \mid y) + s_\rho(y) \overset{\cdot}{\frown} \delta$		CMGC2
$x \mid (\psi :\!\rightarrow y) = \phi :\!\rightarrow (x \mid y) + s_\rho(x) \overset{\cdot}{\frown} \delta$		CMGC3
$\partial_H(a) = a$	if $a \notin H$	D1
$\partial_H(a) = \delta$	if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		D3
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$		D4
$\partial_H(\phi \overset{\cdot}{\frown} x) = \phi \overset{\cdot}{\frown} \partial_H(x)$		DRS
$\partial_H(x \overset{\phi}{\frown}) = \partial_H(x) \overset{\phi}{\frown}$		DTS
$\partial_H(\phi :\!\rightarrow x) = \phi :\!\rightarrow \partial_H(x)$		DGC

TABLE 15. Merge with communication and encapsulation.

We provide the semantics of ACPps in table 16. The semantics of PAps can be extracted, by omitting all parts referring to the communication merge operator.

$$\frac{x \xrightarrow{\phi,a} x',\ s_\rho(x\|y) \neq F,\ s_\rho(x'\|y) \neq F}{x\|y \xrightarrow{\phi,a} x'\|y,\ y\|x \xrightarrow{\phi,a} y\|x'}$$

$$\frac{x \xrightarrow{\phi,a} \psi,\ s_\rho(x\|y) \neq F,\ \psi\wedge s_\rho(y) \neq F}{x\|y \xrightarrow{\phi,a} \psi \lceil y,\ y\|x \xrightarrow{\phi,a} \psi \lceil y}$$

$$\frac{x \xrightarrow{\phi,a} x',\ s_\rho(x'\|y) \neq F}{x\mathbb{L}y \xrightarrow{\phi,a} x'\|y} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ \psi\wedge s_\rho(y) \neq F}{x\mathbb{L}y \xrightarrow{\phi,a} \psi \lceil y}$$

$$\frac{x \xrightarrow{\phi,a} x',\ y \xrightarrow{\psi,a} y',\ s_\rho(x\|y) \neq F,\ s_\rho(x'\|y') \neq F,\ a\,|\,b=c}{x\|y \xrightarrow{\phi\wedge\psi,c} x'\|y',\ x\,|\,y \xrightarrow{\phi\wedge\psi,c} x'\|y'}$$

$$\frac{x \xrightarrow{\phi,a} \chi,\ y \xrightarrow{\psi,a} \xi,\ s_\rho(x\|y) \neq F,\ \chi\wedge\xi \neq F,\ a\,|\,b=c}{x\|y \xrightarrow{\phi\wedge\psi,c} \chi\wedge\xi,\ x\,|\,y \xrightarrow{\phi\wedge\psi,c} \chi\wedge\xi}$$

$$\frac{x \xrightarrow{\phi,a} \chi,\ y \xrightarrow{\psi,a} y',\ s_\rho(x\|y) \neq F,\ \chi\wedge s_\rho(y') \neq F,\ a\,|\,b=c}{\begin{array}{c} x\|y \xrightarrow{\phi\wedge\psi,c} \chi \lceil y',\ y\|x \xrightarrow{\phi\wedge\psi,c} \chi \lceil y' \\ x\,|\,y \xrightarrow{\phi\wedge\psi,c} \chi \lceil y',\ y\,|\,x \xrightarrow{\phi\wedge\psi,c} \chi \lceil y' \end{array}}$$

$$\frac{x \xrightarrow{\phi,a} x',\ a\notin H}{\partial_H(x) \xrightarrow{\phi,a} \partial_H(x')} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ a\notin H}{\partial_H(x) \xrightarrow{\phi,a} \psi}$$

$$s_\rho(x\|y) = s_\rho(x) \wedge s_\rho(y)$$
$$s_\rho(x\mathbb{L}y) = s_\rho(x)$$
$$s_\rho(x\,|\,y) = s_\rho(x) \wedge s_\rho(y)$$
$$s_\rho(\partial_H(x)) = s_\rho(x)$$

TABLE 16. Semantics of ACPps.

4. STATE OPERATOR.

In this section, we extend any of the theories BPAps, PAps, ACPps with the state operator of [BAB88]. The interesting aspect here is, that we allow the state to be (partly) visible to the process, i.e. a state can emit a signal.

4.1. SYNTAX AND SEMANTICS.

Let us assume that a state operator in the sense of [BAB88] is given by a domain S and functions $act: A \times S \to A \cup \{\delta\}$ and $eff: A \times S \to S$. The expression $\lambda_s(x)$ with $s \in S$ denotes process x working on the state space S with the current state being $s \in S$.

We can assume that there is an additional function $sig: S \to B$ which determines for each state the signal that is emitted by that state. The absence of signals is modeled by taking $sig(s) = T$ of course. Now the eight equations for the state operator are as shown in table 17, the operational semantics is given in table 18.

$\lambda_s(\bot) = \bot$	SOS0
$\lambda_s(\delta) = sig(s) \curvearrowright \delta$	SOS1
$\lambda_s(a) = sig(s) \curvearrowright act(a, s) \curvearrowleft sig(eff(a, s)$	SOS2
$\lambda_s(a \cdot x) = sig(s) \curvearrowright act(a, s) \cdot \lambda_{eff(a, s)}(x)$	SOS3
$\lambda_s(x + y) = \lambda_s(x) + \lambda_s(y)$	SOS4
$\lambda_s(\phi \curvearrowright x) = \phi \curvearrowright \lambda_s(x)$	SOS5
$\lambda_s(x \curvearrowleft \phi) = \lambda_s(x) \curvearrowleft \phi$	SOS6
$\lambda_s(\phi :\to x) = sig(s) \curvearrowright \phi :\to \lambda_s(x)$	SOS7

TABLE 17. State operator generating signals.

$$\frac{x \xrightarrow{\phi,a} x',\ sig(s) \wedge s_\rho(x) \neq F,\ sig(eff(a,s)) \wedge s_\rho(x') \neq F,\ act(a,s) = b \neq \delta}{\lambda_s(x) \xrightarrow{\phi,b} \lambda_{eff(a,s)}(x')}$$

$$\frac{x \xrightarrow{\phi,a} \psi,\ sig(s) \wedge s_\rho(x) \neq F,\ sig(eff(a,s)) \wedge \psi \neq F,\ act(a,s) = b \neq \delta}{\lambda_s(x) \xrightarrow{\phi,b} sig(eff(a,s)) \wedge \psi}$$

$$s_\rho(\lambda_s(x)) = s_\rho(x) \wedge sig(s)$$

TABLE 18. Operational semantics.

Using a state operator that generates signals one can define signaling processes in such a way that the equations need not contain any signal at all, thus considerably optimizing the notation. We will illustrate this in two examples.

4.2 EXAMPLE.
Let D be a finite alphabet of data, and let D^* be the collection of finite sequences over D. The empty sequence is denoted by ε and adding an element d to the list x results in σd. The propositional constants are as follows:

on_top(d) for $d \in D$,

empty.

We will assume that these signals are exclusive, i.e. we will assume that the following formula always holds: $\Phi \equiv (\text{empty} \supset \wedge\wedge_{d\in D}\ \text{on\_top}(d)) \wedge \wedge\wedge_{d\in D}\ (\text{on\_top}(d) \supset \wedge\wedge_{e\neq d}\ \text{on\_top}(e))$.
D^* will be the state space for a process that represents a stack over D. The signal function sig is defined by $\text{sig}(\varepsilon) = \text{empty}$, $\text{sig}(\sigma d) = \text{top}(d)$. The atomic actions are:

push_int(d), push(d) for $d \in D$ (the suffix int denotes an *intended* action),

pop_int, pop.

The functions act and eff are given by:

act(push_int(d), σ) = push(d) (the act function transforms an intended action into an actual action),

act(pop_int, σ) = pop,

eff(push_int(d), σ) = σd (the eff function gives the resulting contents of the stack),

eff(pop_int, ε) = ε,

eff(pop_int, σd) = σ.

(For act only those cases are given where act will not lead to δ.). The behavior of a stack over D is given by the following process definition.

$$\text{stack}(D) = \Phi\,\lceil\!\!\blacktriangledown\, \lambda_{\varepsilon}((\sum_{d\in D} \text{push\_int}(d) + \text{pop\_int})^*\delta).$$

4.3 EXAMPLE.
In this example two buffers A and B with data from the finite set D are maintained in the state. Both buffers have length $k > 1$. The process to be defined allows to read data in both buffers in a concurrent mode. For both buffers A and B there is a propositional constant: openA indicates that there is still room in A (likewise for B). When both buffers have been loaded the process C compares the contents of the buffers. The comparison will send value true if the buffers were equal and false otherwise. Thereafter the buffers are made empty again and the process restarts. We will describe the system in a top-down fashion, first explaining the overall architecture and then completing the details.

$$\text{SYSTEM} = \lambda_{\langle\varepsilon,\varepsilon\rangle}(A \parallel B \parallel C).$$

The state consists of a pair of buffers A and B. Initially both are empty. The signals produced by a state $\langle \alpha, \beta \rangle$ are as follows.

$$\begin{aligned} sig(\langle \alpha, \beta \rangle) = \; & openA \wedge openB && \text{if } length(\alpha) < k \text{ and } length(\beta) < k, \\ & \neg openA \wedge openB && \text{if } length(\alpha) = k \text{ and } length(\beta) < k, \\ & openA \wedge \neg openB && \text{if } length(\alpha) < k \text{ and } length(\beta) = k, \\ & \neg openA \wedge \neg openB && \text{if } length(\alpha) = k \text{ and } length(\beta) = k. \end{aligned}$$

The processes A, B, C are defined by:

$$A = (openA :\rightarrow \sum_{d \in D} readA(d))^* \delta$$

$$B = (openB :\rightarrow \sum_{d \in D} readB(d))^* \delta$$

$$C = (\neg openA \wedge \neg openB :\rightarrow comp)^* \delta$$

The next step is to explain the effect function:

$$eff(readA(d), \langle \alpha, \beta \rangle) = \langle \alpha d, \beta \rangle$$
$$eff(readB(d), \langle \alpha, \beta \rangle) = \langle \alpha, \beta d \rangle$$
$$eff(comp, \langle \alpha, \beta \rangle) = \langle \varepsilon, \varepsilon \rangle$$

Finally the action function must be specified:

$$\begin{aligned} act(comp, \langle \alpha, \beta \rangle) = \; & write(true) && \text{if } \alpha = \beta \\ & write(false) && \text{otherwise,} \end{aligned}$$

and the action function is the identity otherwise.

The use of the state operator in this example is hard to avoid because of the parallel reading of data that must be used simultaneously later on. This issue is worked out in [VER90].

5. ABSTRACTION.

We provide axioms for silent step and abstraction in the setting of branching bisimulation of [GLW89].

5.1 ACP$^{\tau}$PS.

The theory ACP$^{\tau}$ps extends ACPps by the addition of a special constant $\tau \notin A$, the silent step, and a unary operator τ_I for each $I \subseteq A$, the abstraction operator. As axioms we have all axioms of ACPps, with now $a,b \in A \cup \{\delta,\tau\}$, plus the additional axioms of table 19 below.

$a\cdot(\phi \rightarrow \tau) = a \leftarrow \phi$		B1S
$x\cdot(s_\rho(y) \leftarrow (\tau\cdot(y + z) + z)) = x\cdot(y + z)$		B2S
$\tau_I(a) = a$	if $a \notin I$	TI1
$\tau_I(a) = \tau$	if $a \in I$	TI2
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$		TI3
$\tau_I(x\cdot y) = \tau_I(x)\cdot\tau_I(y)$		TI4
$\tau_I(\phi \leftarrow x) = \phi \leftarrow \tau_I(x)$		TIRS
$\tau_I(x \leftarrow \phi) = \tau_I(x) \leftarrow \phi$		TITS
$\tau_I(\phi{:}\rightarrow x) = \phi{:}\rightarrow \tau_I(x)$		TIGC

TABLE 19. Silent step and abstraction.

5.2 SEMANTICS.

The operational semantics now also has arrow labels of the form ϕ,τ. In the previous rules, we now have $a \in A \cup \{\tau\}$. The additional rules for the abstraction operator are shown in table 20.

$$\frac{x \xrightarrow{\phi,a} x',\ a \notin I}{\tau_I(x) \xrightarrow{\phi,a} \tau_I(x')} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ a \notin I}{\tau_I(x) \xrightarrow{\phi,a} \psi}$$

$$\frac{x \xrightarrow{\phi,a} x',\ a \in I}{\tau_I(x) \xrightarrow{\phi,\tau} \tau_I(x')} \qquad \frac{x \xrightarrow{\phi,a} \psi,\ a \in I}{\tau_I(x) \xrightarrow{\phi,\tau} \psi}$$

$$s_\rho(\tau_I(x)) = s_\rho(x)$$

TABLE 20. Semantics of $ACP^\tau ps$.

With this comes a new definition of bisimulation. We consider the set $\mathbb{PB}$ of closed terms and propositional formulas. In the following, $x,y,x',y',\ldots$ range over terms, $\phi,\psi,\ldots$ range over propositions and $s,t,\ldots$ over $\mathbb{PB}$.

A relation R on $\mathbb{PB}$ is a *branching bisimulation* when the following holds:

i. if xRy then $s_\rho(x) = s_\rho(y)$, if $xR\phi$ then $s_\rho(x) = \phi$, if ϕRx then $\phi = s_\rho(x)$ and if $\phi R\psi$ then $\phi = \psi$

ii. if xRs and $x \xrightarrow{\phi,a} t$, then either:

a. $a \equiv \tau$, $s_\rho(x) \supset \phi = T$ and tRs, or:

b. for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are propositions $\psi, \psi_1, \ldots, \psi_n$ $(n \geq 0)$ and $s', y_1, \ldots, y_n$ such that $s_\rho(x) \supset \psi_i = T$ for all i, $v(\psi) = T$, $s \xrightarrow{\psi_1,\tau} y_1 \ldots \xrightarrow{\psi_n,\tau} y_n \xrightarrow{\psi,a} s'$ and xRy_i for all i and tRs'

iii. vice versa: if sRx and $x \xrightarrow{\phi,a} t$, then either:

a. $a \equiv \tau$, $s_\rho(x) \supset \phi = T$ and sRt, or:

b. for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are propositions $\psi, \psi_1, ..., \psi_n$ $(n \geq 0)$ and $s', y_1, ..., y_n$ such that $s_\rho(x) \supset \psi_i = T$ for all i, $v(\psi) = T$, $s \xrightarrow{\psi_1,\tau} y_1 \ ... \ \xrightarrow{\psi_n,\tau} y_n \xrightarrow{\psi,a} s'$ and y_iRx for all i and $s'Rt$.

We say a branching bisimulation R satisfies the *root condition* for x and y if xRy and in addition:

iv. if $x \xrightarrow{\phi,a} t$, then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there is a proposition ψ and a term or proposition s such that $v(\psi) = T$, $y \xrightarrow{\psi,a} s$ and tRs

v. vice versa: if $y \xrightarrow{\phi,a} s$, then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there is a proposition ψ and a term or proposition t such that $v(\psi) = T$, $x \xrightarrow{\psi,a} t$ and tRs.

We call two expressions x,y *branching bisimilar*, notated $x \leftrightarrows_b y$, if there is a branching bisimulation relating x and y. Two expressions x,y are *rooted branching bisimilar*, $x \leftrightarrows_{rb} y$, if there is a branching bisimulation that satisfies the root condition for x and y.

5.3 THEOREM. For all closed $ACP^\tau ps$ terms t,s we have

$$ACP^\tau ps \vdash t = s \ \Leftrightarrow \ \mathbb{PB}/\leftrightarrows_{rb} \models t = s,$$

i.e. $ACP^\tau ps$ is a sound and complete axiomatisation of the model $\mathbb{PB}/\leftrightarrows_{rb}$.

PROOF: Similar to the standard proof of the completeness of ACP^τ (see [BAW90]).

6. EXAMPLES.

In this section we discuss a number of examples of the use of signals and inspection.

6.1 QUEUE.

A specification of a (FIFO) queue can be given as follows. We have a given finite data set D, and the following specifications have variables indexed by sequences over D.

$$Q_\varepsilon = \text{empty}^{\wedge} \sum_{d \in D} r(d) \cdot Q_d$$

$$Q_{\sigma d} = \neg\text{empty}^{\wedge} s(d) \cdot Q_\sigma + \sum_{e \in D} r(e) \cdot Q_{e\sigma d}.$$

6.2 BAG.

The bag is indexed by multi-sets.

$$B_\varnothing = \text{empty}^{\wedge} \sum_{d \in D} r(d) \cdot B_{\{d\}}$$

$$V \neq \varnothing \Rightarrow B_V = \neg\text{empty}^{\wedge} \sum_{d \in V} s(d) \cdot B_{V-\{d\}} + \sum_{d \in D} r(d) \cdot B_{V \cup \{d\}}.$$

6.3 STACK.

We use conventions as above. We give a number of alternatives. First a stack without signals.

$$S^1_\varepsilon = \sum_{d\in D} push(d)\cdot S^1_d$$

$$S^1_{\sigma d} = pop\cdot S^1_\sigma + top(d)\cdot S^1_\sigma + \sum_{e\in D} push(e)\cdot S^1_{\sigma de}.$$

Next, we add a signal showing the top of the stack.

$$S^2_\varepsilon = \sum_{d\in D} push(d)\cdot S^2_d$$

$$S^2_{\sigma d} = pop\cdot S^2_\sigma + top\cdot(show(d) \leftsquigarrow S^2_{\sigma d}) + \sum_{e\in D} push(e)\cdot S^2_{\sigma de}.$$

In the third specification, we add a signal empty, and also allow actions top, pop in case the stack is empty. If this happens, an error signal is emitted, and no further action is possible.

$$S^3_\varepsilon = empty \leftsquigarrow \sum_{d\in D} push(d)\cdot S^3_d + top \leftarrowtail error + pop \leftarrowtail error$$

$$S^3_{\sigma d} = pop\cdot S^3_\sigma + top\cdot(show(d) \leftsquigarrow S^3_{\sigma d}) + \neg empty \leftsquigarrow \sum_{e\in D} push(e)\cdot S^3_{\sigma de}.$$

The fourth specification has a state of underflow, when an empty stack is popped. A subsequent push leads out of the error situation.

$$S^4_\varepsilon = empty \leftsquigarrow \sum_{d\in D} push(d)\cdot S^4_d + top \leftarrowtail error + pop\cdot U^4$$

$$U^4 = underflow \leftsquigarrow \sum_{d\in D} push(d)\cdot S^4_\varepsilon + top \leftarrowtail error + pop\cdot U^4$$

$$S^4_{\sigma d} = pop\cdot S^4_\sigma + top\cdot(show(d) \leftsquigarrow S^4_{\sigma d}) + \neg empty \leftsquigarrow \sum_{e\in D} push(e)\cdot S^4_{\sigma de}.$$

The fifth stack keeps functioning, when a pop or top is executed on an empty stack.

$$S^5_\varepsilon = empty \leftsquigarrow \sum_{d\in D} push(d)\cdot S^5_d + top\cdot(show(\perp) \leftsquigarrow S^5_\varepsilon) + pop\cdot(error \leftsquigarrow S^5_\varepsilon)$$

$$S^5_{\sigma d} = pop\cdot S^5_\sigma + top\cdot(show(d) \leftsquigarrow S^5_{\sigma d}) + \neg empty \leftsquigarrow \sum_{e\in D} push(e)\cdot S^5_{\sigma de}.$$

In the sixth specification, a pop or top executed on an empty stack leads to an irrecoverable error state, but actions can still be executed.

$$S^6_\varepsilon = empty \leftsquigarrow \sum_{d\in D} push(d)\cdot S^6_d + (top + pop)\cdot(error \leftsquigarrow (top + pop + \sum_{d\in D} push(d))^*\delta)$$

$$S^6_{\sigma d} = pop \cdot S^6_{\sigma} + top \cdot (show(d) \triangleleft S^6_{\sigma d}) + \neg empty \triangleleft \sum_{e \in D} push(e) \cdot S^6_{\sigma d e}.$$

6.4 COMMUNICATING BUFFERS.

In this example we study a system where both signal inspection and communication play a role. We will show that communication can be replaced by inspection. We start out from a standard specification of one element buffers, that in addition always signal the contents on the output port. The buffer B^{ij} has input port i and output port j, and can buffer messages from some finite set D. Let $\varnothing \notin D$. The signal $show_j(d)$ means that message d is offered at port j ($d \in D$), $show_j(\varnothing)$ means that the buffer is empty.

$$B^{ij} = show_j(\varnothing) \triangleleft \sum_{d \in D} read_i(d) \cdot B^{ij}_d$$

$$B^{ij}_d = show_j(d) \triangleleft send_j(d) \cdot B^{ij}.$$

$$X = \partial_H(B^{12} \parallel B^{23})$$

where $send_2(d) \mid read_2(d) = comm_2(d)$ (communication gives δ otherwise),
and $H = \{read_2(d), send_2(d) : d \in D\}$.
Some calculations result in the following recursive specification:

$$X = (show_2(\varnothing) \wedge show_3(\varnothing)) \triangleleft \sum_{d \in D} read_1(d) \cdot X^d_1$$

$$X^d_1 = (show_2(d) \wedge show_3(\varnothing)) \triangleleft comm_2(d) \cdot X^d_2$$

$$X^d_2 = (show_2(\varnothing) \wedge show_3(d)) \triangleleft send_3(d) \cdot X + \sum_{e \in D} read_1(d) \cdot X^{de}_3$$

$$X^{de}_3 = (show_2(e) \wedge show_3(d)) \triangleleft send_3(d) \cdot X^e_1$$

Hiding all signals gives back the usual specification of two coupled one-element buffers (as in [BAW90], page 106).

As a first step in replacing communication by inspection, we omit the parametrisation of the communication action in favor of signal inspection. To make this correct, we need to require that signals are exclusive, formalised by proposition

$$\Phi_i \equiv (show_i(\varnothing) \supset \bigwedge_{d \in D} \neg show_i(d)) \wedge \bigwedge_{d \in D} (show_i(d) \supset \neg show_i(\varnothing) \wedge \bigwedge_{e \neq d} \neg show_i(e)).$$

$$C^{ij} = show_j(\varnothing) \triangleleft \sum_{d \in D} show_i(d) :\rightarrow read_i \cdot C^{ij}_d$$

$$C^{ij}_d = show_j(d) \triangleleft send_j \cdot C^{ij}.$$

$$Y = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \triangleright \partial_H(C^{12} \parallel C^{23}),$$

where communication is given by $send_2 \mid read_2 = comm_2$,
and encapsulation by $H = \{read_2, send_2\}$.
Some calculations result in the following recursive specification (omitting the exclusivity propositions):

$$Y = (show_2(\varnothing) \wedge show_3(\varnothing)) \leadsto \sum_{d \in D} show_1(d) :\rightarrow read_1 \cdot Y_1^d$$

$$Y_1^d = (show_2(d) \wedge show_3(\varnothing)) \leadsto comm_2 \cdot Y_2^d$$

$$Y_2^d = (show_2(\varnothing) \wedge show_3(d)) \leadsto send_3 \cdot Y + \sum_{e \in D} show_1(e) :\rightarrow read_1 \cdot Y_3^{de}$$

$$Y_3^{de} = (show_2(e) \wedge show_3(d)) \leadsto send_3 \cdot Y_1^e$$

Let us now abstract from actions and signals at port 2. Put $I = \{comm_2\}$ and $show_2 = \{show_2(d) : d \in D\} \cup \{show_2(\varnothing)\}$, and derive the following specification for $Z = show_2 \Delta \tau_I(Y)$:

$$Z = show_2 \Delta \tau_I(Y) = show_3(\varnothing) \leadsto \sum_{d \in D} show_1(d) :\rightarrow read_1 \cdot Z_1^d$$

$$Z_1^d = show_2 \Delta \tau_I(Y_1^d) = show_3(\varnothing) \leadsto \tau \cdot Z_2^d$$

$$Z_2^d = show_2 \Delta \tau_I(Y_2^d) = show_3(d) \leadsto send_3 \cdot Z + \sum_{e \in D} show_1(e) :\rightarrow read_1 \cdot Z_3^{de}$$

$$Z_3^{de} = show_2 \Delta \tau_I(Y_3^{de}) = show_3(d) \leadsto send_3 \cdot Z_1^e$$

Next, we can do away with the synchronisation in favour of two extra signals at the connecting port. First, we consider the specification without extra actions:

$$E^{ij} = (show_j(d) \wedge \neg flag_i) \leadsto \sum_{d \in D} (show_i(d) \wedge \neg flag_j) :\rightarrow read_i \cdot E_d^{ij}$$

$$E_d^{ij} = (show_j(d) \wedge flag_i) \leadsto (show_i(\varnothing) \wedge \neg flag_j) :\rightarrow s_j \cdot C^{ij}.$$

$$W = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \lceil\!\rceil E^{12} \| E^{23},$$

where this is the free merge, i.e. this is a specification in PAps.

Unfortunately, this system does not behave as a two-item buffer but as a one-item buffer. If we want the intended behaviour, we have to put in extra actions:

$$F^{ij} = (ready_i \wedge show_j(\varnothing) \wedge \neg flag_j) \leadsto \sum_{d \in D} (flag_i \wedge show_i(d)) :\rightarrow read_i \cdot F_d^{ij}$$

$$F_d^{ij} = (\neg ready_i \wedge show_j(d) \wedge \neg flag_j) \leadsto (ready_j \wedge \neg flag_1) :\rightarrow send_j \cdot G_d^{ij}.$$

$$G_d^{ij} = (\neg ready_i \wedge show_j(d) \wedge flag_j) \leadsto \neg ready_j :\rightarrow reset_j \cdot G_d^{ij}.$$

$$V = (\Phi_1 \wedge \Phi_2 \wedge \Phi_3) \lceil\!\rceil F^{12} \| F^{23},$$

where this is the free merge, i.e. this is a specification in PAps.

We can derive the following specification for V:

$$V = (ready_1 \wedge ready_2 \wedge show_2(\varnothing) \wedge show_3(\varnothing) \wedge \neg flag_2 \wedge \neg flag_3)^{\frown} \sum_{d \in D} (flag_1 \wedge show_1(d)) :\rightarrow read_1 \cdot V_{1,d}$$

$$V_{1,d} = (\neg ready_1 \wedge ready_2 \wedge show_2(d) \wedge show_3(\varnothing) \wedge \neg flag_2 \wedge \neg flag_3)^{\frown} \neg flag_1 :\rightarrow send_2 \cdot V_{2,d}$$

$$V_{2,d} = (\neg ready_1 \wedge ready_2 \wedge show_2(d) \wedge show_3(\varnothing) \wedge flag_2 \wedge \neg flag_3)^{\frown} read_2 \cdot V_{3,d}$$

$$V_{3,d} = (\neg ready_1 \wedge \neg ready_2 \wedge show_2(d) \wedge show_3(d) \wedge flag_2 \wedge \neg flag_3)^{\frown} reset_2 \cdot V_{4,d}$$

$$V_{4,d} = (ready_1 \wedge \neg ready_2 \wedge show_2(\varnothing) \wedge show_3(d) \wedge \neg flag_2 \wedge \neg flag_3)^{\frown} ready_3 :\rightarrow send_3 \cdot V_{5,d} + \sum_{e \in D} (flag_1 \wedge show_1(e)) :\rightarrow read_1 \cdot V_{6,de}$$

$$V_{5,d} = (ready_1 \wedge \neg ready_2 \wedge show_2(\varnothing) \wedge show_3(d) \wedge \neg flag_2 \wedge flag_3)^{\frown} \neg ready_3 :\rightarrow reset_3 \cdot V + \sum_{e \in D} (flag_1 \wedge show_1(e)) :\rightarrow read_1 \cdot V_{7,de}$$

$$V_{6,de} = (\neg ready_1 \wedge \neg ready_2 \wedge show_2(e) \wedge show_3(d) \wedge \neg flag_2 \wedge \neg flag_3)^{\frown} ready_3 :\rightarrow send_3 \cdot V_{7,de}$$

$$V_{7,de} = (\neg ready_1 \wedge \neg ready_2 \wedge show_2(e) \wedge show_3(d) \wedge \neg flag_2 \wedge flag_3)^{\frown} \neg ready_3 :\rightarrow reset_3 \cdot V_{1,e}$$

Now we hide all signals at port 2, and the additional signals introduced in the last step, i.e. we hide the propositional constants in the set

$$P = \{show_2(d) : d \in D\} \cup \{show_2(\varnothing), ready_1, ready_2, ready_3, flag_1, flag_2, flag_3\}$$

We obtain the following specification for $P \Delta V$:

$$P \Delta V = show_3(\varnothing)^{\frown} \sum_{d \in D} show_1(d) :\rightarrow read_1 \cdot (P \Delta V_{1,d})$$

$$P \Delta V_{1,d} = show_3(\varnothing)^{\frown} send_2 \cdot (P \Delta V_{2,d})$$

$$P \Delta V_{2,d} = show_3(\varnothing)^{\frown} read_2 \cdot (P \Delta V_{3,d})$$

$$P \Delta V_{3,d} = show_3(d)^{\frown} reset_2 \cdot (P \Delta V_{4,d})$$

$$P \Delta V_{4,d} = show_3(d)^{\frown} send_3 \cdot (P \Delta V_{5,d}) + \sum_{e \in D} show_1(e) :\rightarrow read_1 \cdot (P \Delta V_{6,de})$$

$$P \Delta V_{5,d} = show_3(d)^{\frown} reset_3 \cdot (P \Delta V) + \sum_{e \in D} show_1(e) :\rightarrow read_1 \cdot (P \Delta V_{7,de})$$

$$P \Delta V_{6,de} = show_3(d)^{\frown} send_3 \cdot (P \Delta V_{7,de})$$

$$P \Delta V_{7,de} = show_3(d)^{\frown} reset_3 \cdot (P \Delta V_{1,e})$$

As the next step, we abstract from the action set $I = \{read_2, send_2, comm_2, reset_2, reset_3\}$.

We obtain:

$$\tau_I(P \Delta V) = show_3(\varnothing)^{\frown} \sum_{d \in D} show_1(d) :\rightarrow read_1 \cdot \tau_I(P \Delta V_{1,d})$$

$\tau_I(P \Delta V_{1,d}) = show_3(\varnothing)\frown \tau\cdot\tau_I(P \Delta V_{2,d})$

$\tau_I(P \Delta V_{2,d}) = show_3(\varnothing)\frown \tau\cdot\tau_I(P \Delta V_{3,d})$

$\tau_I(P \Delta V_{3,d}) = show_3(d)\frown \tau\cdot\tau_I(P \Delta V_{4,d})$

$\tau_I(P \Delta V_{4,d}) = show_3(d)\frown send_3\cdot\tau_I(P \Delta V_{5,d}) + \sum_{e\in D} show_1(e):\rightarrow read_1\cdot\tau_I(P \Delta V_{6,de})$

$\tau_I(P \Delta V_{5,d}) = show_3(d)\frown \tau\cdot\tau_I(P \Delta V) + \sum_{e\in D} show_1(e):\rightarrow read_1\cdot\tau_I(P \Delta V_{7,de})$

$\tau_I(P \Delta V_{6,de}) = show_3(d)\frown send_3\cdot\tau_I(P \Delta V_{7,de})$

$\tau_I(P \Delta V_{7,de}) = show_3(d)\frown \tau\cdot\tau_I(P \Delta V_{1,e})$

Using the laws for branching bisimulation, we can reduce this to:

$\tau_I(P \Delta V) = show_3(\varnothing)\frown \sum_{d\in D} show_1(d):\rightarrow read_1\cdot\tau_I(P \Delta V_{2,d})$

$\tau_I(P \Delta V_{2,d}) = show_3(\varnothing)\frown \tau\cdot\tau_I(P \Delta V_{4,d})$

$\tau_I(P \Delta V_{4,d}) = show_3(d))\frown send_3\cdot\tau_I(P \Delta V) + \sum_{e\in D} show_1(e):\rightarrow read_1\cdot\tau_I(P \Delta V_{6,de})$

$\tau_I(P \Delta V_{6,de}) = show_3(d))\frown send_3\cdot\tau_I(P \Delta V_{1,e})$

and we see that this is the same specification as for Z above.

7. CONCLUSION.

We conclude that we have described the interplay between the execution of actions of a process (giving the state changes, the dynamics of a process) and the propositions that hold in a state of a process (giving the static part of a process). The signal emitted by a state is a proposition that consitutes the visible part of this state, and an action leading out of a state can be conditional on a proposition that should hold in a state. In a parallel composition of two processes, an action executed by one process can be conditional, depending on the signal emitted by the other process. This described a mechanism called signal inspection or signal observation.

We have given some small examples. Further work would be to construct larger examples, and to extend both logic and process theory, for instance with timing constructs.

REFERENCES.

[AUB84] D. AUSTRY & G. BOUDOL, *Algèbre de processus et synchronisation,* Theoretical Computer Science 30, 1984, pp. 91-131.

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

[BAB92] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with signals and conditions,* in: Programming and mathematical method, Proc. Summer School, Marktoberdorf 1990 (M. Broy, ed.), NATO ASI Series F 88, Springer Verlag 1992, pp. 273-323.

[BAV93] J.C.M. BAETEN & C. VERHOEF, *A congruence theorem for structured operational semantics with predicates,* in Proc. CONCUR'93, Hildesheim (E. Best, ed.), Springer LNCS 715, 1993, pp. 477-492.

[BAV94] J.C.M. BAETEN & C. VERHOEF, *Concrete process algebra,* to appear in: Handbook of logic in computer science (S. Abramsky, D.M. Gabbay & T.S.E. Maibaum, eds.), Vol. IV, Syntactical Methods, Chapter 2, pp. 149 - 268.

[BAW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press 1990.

[BEBP94] J.A. BERGSTRA, I. BETHKE & A. PONSE, *Process algebra with iteration and nesting,* The Computer Journal 37, 1994, pp. 243-258.

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

[BEP94] J.A. BERGSTRA & A. PONSE, *Frame based process logics,* to appear in Proc. Bisim-3, CSLI.

[BRHR84] S.D. BROOKES, C.A.R. HOARE & W. ROSCOE, *A theory of communicating sequential processes,* Journal of the ACM 31, pp. 560-599.

[BRO90] W.S. BROUWER, *Stable signals and observation in a process specification formalism,* M.Sc. Thesis, University of Amsterdam 1990.

[GLW89] R.J. VAN GLABBEEK & W.P. WEIJLAND, *Branching time and abstraction in bisimulation semantics. Extended abstract,* in: Information Processing 89, IFIP World Congress, San Francisco 1989 (G.X. Ritter, ed.), North-Holland, Amsterdam 1989, pp. 613-618.

[GRP94] J.F. GROOTE & A. PONSE, *Process algebra with guards (combining Hoare logic with process algebra),* Formal Aspects of Computing 6, 1994, pp. 115-164.

[NIS94] X. NICOLLIN & J. SIFAKIS, *The algebra of timed processes ATP: theory and application,* Information & Computation 114, 1994, pp. 131-178.

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.

91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.
92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.

92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach

		Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness?, p. 24.
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.

93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.
94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doombos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.

94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving ∀CTL*, ∃CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.

94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.
94/39	A. Blokhuis T. Kloks	On the equivalence covering number of splitgraphs, p. 4.
94/40	D. Alstein	Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch	Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken	Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld	Category Theory as Coherently Constructive Lattice M . Theory: An Illustration, p. 35.
94/44	E. Brinksma J. Davies R. Gerth S. Graf W. Janssen B. Jonsson S. Katz G. Lowe M. Poel A. Pnueli C. Rump J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben	Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt	The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt	On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group	Fixed-Point Calculus, p. 11.