Process algebra with propositional signals

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.
Note: This research was supported in part by ESPRIT basic research action 7166, CONCUR2.

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 conditional 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:

$$\boxed{\text{thevisiblepart(signal)ofthestateofaprocessisaproposition}}.$$

Whatever the merits of this position, what we do establish is that it is a consistent one, 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 due to the different models, based on failure or ready sets.

ACKNOWLEDGEMENT: Helpful remarks by Jan Joris Vereijken (Eindhoven University of Technology) and Joris Boselie (University of Amsterdam) are appreciated.

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	$x + \delta = x$	A6
$(x + y) + z = x + (y + z)$	A2	$\delta \cdot x = \delta$	A7
$x + x = x$	A3	$x + \bot = \bot$	NE1
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$\bot \cdot x = \bot$	NE2
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$a \cdot \bot = \delta$	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 $P_1, \ldots, P_n$ 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].

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

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. $\phi^{\nearrow}$ is the root signal emission operator and $\phi^{\nwarrow}$ is the terminal signal emission operator (we take this notation from [BEP94]). 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).

$(\phi \, \searrow x) \cdot y = \phi \, \searrow (x \cdot y)$	RSE1	$F^{\nearrow} x = \perp$	RSE5
$(\phi \, \searrow x) + y = \phi \, \searrow (x + y)$	RSE2	$\phi^{\nearrow} \perp = \perp$	RSE6
$\phi \, \searrow (\psi^{\nearrow} x) = (\phi \wedge \psi)^{\nearrow} x$	RSE3	$\phi :\to (\psi^{\nearrow} x) = (\phi \supset \psi)^{\nearrow} (\phi :\to x)$	RSE7
$T \, \searrow x = x$	RSE4	$\phi^{\nearrow} (\phi :\to x) = \phi^{\nearrow} x$	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:

$$(\phi^{\nearrow} x) + (\psi^{\nearrow} y) = (\phi \wedge \psi)^{\nearrow} (x + y).$$

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:

$$a \cdot ((\phi^{\nearrow} b) + (\neg\phi^{\nearrow} b)) = a \cdot ((\phi \wedge \neg\phi)^{\nearrow} b) = a \cdot (F^{\nearrow} b) = a \cdot \perp = \delta.$$

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:

$$\phi \, \searrow (\psi :\to x) = \phi \, \searrow (\phi :\to (\psi :\to x)) = \phi \, \searrow ((\phi \wedge \psi) :\to x).$$

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	$a \leftarrow F = \delta$	TSE5
$(x + y) \leftarrow \phi = x \leftarrow \phi + y \leftarrow \phi$	TSE2	$\delta \leftarrow \phi = \delta$	TSE6
$(x \leftarrow \phi) \leftarrow \psi = x \leftarrow (\phi \wedge \psi)$	TSE3	$\bot \leftarrow \phi = \bot$	TSE7
$x \leftarrow T = x$	TSE4	$(\phi :\rightarrow x) \leftarrow \psi = \phi :\rightarrow (x \leftarrow \psi)$	TSE8
$(x \leftarrow \phi) \cdot y = x \cdot (\phi \nwarrow y)$	TRSE1	$(\phi \nwarrow x) \leftarrow \psi = \phi \nwarrow (x \leftarrow \psi)$	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. $\phi \in B - \{F\} \Rightarrow \phi \nwarrow \delta \in \mathcal{B}$

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:

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

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

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.

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].

$$a \xrightarrow{T,a} T \qquad \frac{x \xrightarrow{\phi,a} x'}{x \cdot y \xrightarrow{\phi,a} x' \cdot y} \qquad \frac{x \xrightarrow{\phi,a} \psi, \psi \wedge s_\rho \qquad (y) \neq F}{x \cdot y \xrightarrow{\phi,a} \psi \overset{\frown}{} y}$$

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

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

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

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

TABLE 5. SOS rules.

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

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 $v(s_\rho \quad (x) \wedge \phi) = T$, 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 $v(s_\rho \quad (y) \wedge \phi) = T$, there is a condition ψ and an expression x' 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 $x \leftrightarrows y$, if there is a (strong) bisimulation relating x and y.

2.7 PROPOSITION. Bisimulation is an congruence relation on process expressions.
As a consequence, we can consider the algebraic structure $\mathbb{P}/\leftrightarrows$ of process expressions modulo bisimulation equivalence.

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

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 $\phi :\rightarrow a \mathbin{\curlywedge} \psi$ is a summand of t

2.10 THEOREM (COMPLETENESS). Let t,s be two closed BPAps terms. Then $x \leftrightarrows y$ implies BPAps $\vdash t=s$. As a corrolary, we have $\mathbb{P}/\leftrightarrows \models t=s \Leftrightarrow$ 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. With the help of the global signal emission operator, we can define a notion of invariance:
DEFINITION: ϕ is an *invariant* of x if $\phi \mathbin{\curlywedge} x = \phi \mathbin{\overline{\curlywedge}} x$.

$\phi \mathbin{\overline{\curlywedge}} \bot = \bot$	GSE0	$\phi \mathbin{\overline{\curlywedge}} (\psi \mathbin{\curlywedge} x) = \psi \mathbin{\curlywedge} (\phi \mathbin{\overline{\curlywedge}} x)$	GSE4
$\phi \mathbin{\overline{\curlywedge}} a = \phi \mathbin{\curlywedge} a \mathbin{\curlywedge} \phi$	GSE1	$\phi \mathbin{\overline{\curlywedge}} (x \mathbin{\curlywedge} \psi) = (\phi \mathbin{\overline{\curlywedge}} x) \mathbin{\curlywedge} \psi$	GSE5
$\phi \mathbin{\overline{\curlywedge}} (x + y) = (\phi \mathbin{\overline{\curlywedge}} x) + (\phi \mathbin{\overline{\curlywedge}} y)$	GSE2	$\phi \mathbin{\overline{\curlywedge}} (\psi :\rightarrow x) = \psi :\rightarrow (\phi \mathbin{\overline{\curlywedge}} x) +$	
$\phi \mathbin{\overline{\curlywedge}} (x \cdot y) = (\phi \mathbin{\overline{\curlywedge}} x) \cdot (\phi \mathbin{\overline{\curlywedge}} y)$	GSE3	$+ \neg\psi :\rightarrow (\phi \mathbin{\curlywedge} \delta)$	GSE6

TABLE 7. Global signal emission.

$$\frac{x \xrightarrow{\phi,a} x', \psi \wedge s_\rho \quad (x) \neq F}{\psi \mathbin{\overline{\curlywedge}} x \xrightarrow{\phi,a} \psi \mathbin{\overline{\curlywedge}} x'} \qquad \frac{x \xrightarrow{\phi,a} \chi, \psi \wedge s_\rho \quad (x) \neq F, \chi \wedge \psi \neq F}{\psi \mathbin{\overline{\curlywedge}} x \xrightarrow{\phi,a} \psi \wedge \chi} \qquad s_\rho(\phi \mathbin{\overline{\curlywedge}} x) = \phi \wedge s_\rho(x)$$

TABLE 8. Operational semantics of global signal emission.

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 $s_\rho(x) = \top$ 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	$p_\rho(x\cdot y) = p_\rho(x)\cdot y$	RSD3
$p_\rho(a) = a$	RSD1	$p_\rho(\phi \barwedge x) = (\phi{:}\to p_\rho(x)) + (\neg\phi{:}\to\bot)$	RSD4
$p_\rho(x + y) = s_\rho(x+y) :\to$		$p_\rho(x \barwedge \phi) = p_\rho(x) \barwedge \phi$	RSD5
$(p_\rho(x) + p_\rho(y)) + \neg s_\rho(x+y){:}\to\bot$	RSD2	$p_\rho(\phi :\to x) = \phi :\to p_\rho(x)$	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. Example: $P \Delta (a\cdot(P \barwedge b) + a\cdot(\neg P \barwedge b)) = a\cdot(T \barwedge (T{:}\to b)) + a\cdot(T \barwedge (T{:}\to b)) = a\cdot b$. 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]).

$P \Delta \bot = \bot$ SH0

$P \Delta (\phi \barwedge \delta) = (\phi[T/P] \vee \phi[F/P]) \barwedge \delta$ SH1

$P \Delta (x + y) = P \Delta (s_\rho(x+y) \barwedge x) + P \Delta (s_\rho(x+y) \barwedge y)$ SH2

$P \Delta (\phi \barwedge \psi{:}\to a\cdot x) = (\phi[T/P] \vee \phi[F/P]) \barwedge ((\phi\wedge\psi)[T/P]{:}\to a\cdot(P \Delta x) + (\phi\wedge\psi)[F/P]{:}\to a\cdot(P \Delta x))$ SH3

$P \Delta (\phi \barwedge \psi{:}\to a \barwedge \chi) = (\phi[T/P] \vee \phi[F/P]) \barwedge ((\phi\wedge\psi)[T/P]{:}\to a \barwedge (\chi[T/P] \vee \chi[F/P])) + (\phi\wedge\psi)[F/P]{:}\to a \barwedge (\chi[T/P] \vee \chi[F/P]))$ SH4

TABLE 10. Signal hiding.

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

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

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

TABLE 11. Operational semantics of signal hiding.

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

TABLE 12. Binary Kleene Star.

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

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

TABLE 13. Operational semantics of binary Kleene star.

2.15 LINEAR RECURSION.

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

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

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 $\overline{\triangledown}$, $\|$, $\between$, and the axioms of tables 7 and 14. Note: $\bot \between x = (F \leftsquigarrow a) \between x = F \leftsquigarrow (a \between x) = \bot$, and $a \between x = (a \blacktriangle T) \between x = a \cdot (T \overline{\triangledown} x)$, where the last expression can be proven equal to $a \cdot x$ for all closed terms.

$x \| y = x \between y + y \between x$	M1	$(x + y) \between z = x \between z + y \between z$	M4
$(a \blacktriangle \phi) \between x = a \cdot (\phi \overline{\triangledown} x)$	M2TS	$(\phi \leftsquigarrow x) \between y = \phi \leftsquigarrow (x \between y)$	MRS
$a \cdot x \between y = a \cdot (x \| y)$	M3	$(\phi :\rightarrow x) \between y = \phi :\rightarrow (x \between y)$	MGC

TABLE 14. Free merge.

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}.

$TL(green) = (green \wedge \neg yellow \wedge \neg red) \leftsquigarrow change \cdot TL(yellow)$

$TL(yellow) = (\neg green \wedge yellow \wedge \neg red) \leftsquigarrow change \cdot TL(red)$

$TL(red) = (\neg green \wedge \neg yellow \wedge red) \leftsquigarrow change \cdot TL(green).$

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 $|$, ∂_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. 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.

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

TABLE 15. Merge with communication and encapsulation.

$$\frac{x \xrightarrow{\phi,a} x', s_\rho \quad (x \between y) \neq F, s_\rho \quad (x' \between y) \neq F}{x \between y \xrightarrow{\phi,a} x' \between y, y \between x \xrightarrow{\phi,a} \rightarrow y \between x'} \qquad \frac{x \xrightarrow{\phi,a} \psi, s_\rho \quad (x \between y) \neq F, \psi \wedge s_\rho \quad (y) \neq F}{x \between y \xrightarrow{\phi,a} \psi \quad \rightharpoondown y, y \between x \xrightarrow{\phi,a} \psi \quad \rightharpoondown y}$$

$$\frac{x \xrightarrow{\phi,a} x', s_\rho \quad (x' \between y) \neq F}{x \lfloor\!\lfloor y \xrightarrow{\phi,a} x' \between y} \qquad \frac{x \xrightarrow{\phi,a} \psi, \psi \wedge s_\rho \quad (y) \neq F}{x \lfloor\!\lfloor y \xrightarrow{\phi,a} \psi \quad \rightharpoondown y} \qquad s_\rho(x \between y) = s_\rho(x) \wedge s_\rho(y)$$

$$\frac{x \xrightarrow{\phi,a} x', y \xrightarrow{\psi,b} \rightarrow y', s_\rho \quad (x \between y) \neq F, s_\rho \quad (x' \between y') \neq F, a \mid b = c}{x \between y \xrightarrow{\phi \wedge \psi,} \overset{c}{x'} \between y', x \mid y \xrightarrow{\phi \wedge \psi,} \rightarrow \overset{c}{x'} \between y'} \qquad s_\rho(x \lfloor\!\lfloor y) = s_\rho(x)$$

$$\frac{x \xrightarrow{\phi,a} \chi, y \xrightarrow{\psi,b} \rightarrow \xi, s_\rho \quad (x \between y) \neq F, \chi \wedge \xi \neq F, a \mid b = c}{x \between y \xrightarrow{\phi \wedge \psi,} \overset{c}{\chi \wedge \xi}, x \mid y \xrightarrow{\phi \wedge \psi,} \rightarrow \overset{c}{\chi \wedge \xi}} \qquad s_\rho(x \mid y) = s_\rho(x) \wedge s_\rho(y)$$

$$\frac{x \xrightarrow{\phi,a} \chi, y \xrightarrow{\psi,b} \rightarrow y', s_\rho \quad (x \between y) \neq F, \chi \wedge s_\rho \quad (y') \neq F, a \mid b = c}{x \between y \xrightarrow{\phi \wedge \psi,c} \chi \quad \rightharpoondown y', y \between x \xrightarrow{\phi \wedge \psi,c} \chi \quad \rightharpoondown y'} \qquad x \mid y \xrightarrow{\phi \wedge \psi c} \chi \quad \rightharpoondown y', y \mid x \xrightarrow{\phi \wedge \psi c} \chi \quad \rightharpoondown y'$$

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

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.

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 a simple example.

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

TABLE 17. State operator generating signals.

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

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

TABLE 18. Operational semantics.

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 $d \in D$, and $empty$.

We will assume that these signals are exclusive, i.e. we will assume that the following formula always holds: $\Phi \equiv (empty \supset \wedge\wedge_{d \in D}\, on\_top(d)) \wedge \wedge\wedge_{d \in D}\, (on\_top(d) \supset \wedge\wedge_{e \neq d}\, on\_top(e))$.

D^* will be the state space for a process that represents a stack over D. The signal function sig is defined by $sig(\varepsilon) = empty$, $sig(\sigma d) = top(d)$. 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:

$act(push\_int(d), \sigma) = push(d)$ (the act function transforms an intended action into an actual

$act(pop\_int, \sigma) = pop$, action),

$eff(push\_int(d), \sigma) = \sigma d$ (the eff function gives the resulting contents of the stack),

$eff(pop\_int, \varepsilon) = \varepsilon$,

$eff(pop\_int, \sigma d) = \sigma$.

(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.

$$stack(D) = \Phi^{\overline{\ }} \lambda_\varepsilon((\sum_{d \in D} push\_int(d) + pop\_int)^* \delta).$$

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.

$x \cdot (s_\rho(y) \mathbin{\overset{\frown}{\blacktriangleleft}} (\tau \cdot (y + z) + z)) = x \cdot (y + z)$		BS	$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	TI4
$\tau_I(a) = a$	if $a \notin I$	TI1	$\tau_I(\phi \mathbin{\overset{\frown}{\blacktriangleleft}} x) = \phi \mathbin{\overset{\frown}{\blacktriangleleft}} \tau_I(x)$	TIRS
$\tau_I(a) = \tau$	if $a \in I$	TI2	$\tau_I(x \mathbin{\overset{\frown}{\blacktriangleright}} \phi) = \tau_I(x) \mathbin{\overset{\frown}{\blacktriangleright}} \phi$	TITS
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$		TI3	$\tau_I(\phi :\rightarrow x) = \phi :\rightarrow \tau_I(x)$	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.

With this comes a new definition of bisimulation. In the following, x,y,x',y',... range over terms and ϕ,ψ,... range over propositions.

A relation R on process expressions is a *branching bisimulation* when the following holds:

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

ii. if xRy and $x \overset{\phi,a}{\rightarrow} x'$, then either:

a. $a \equiv \tau$, $s_\rho(x) \supset \phi = T$ and $x'Ry$, or:

b. for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are propositions ψ, ψ_1, ..., ψ_n ($n \geq 0$) and $y', y_1, ..., y_n$ such that $s_\rho(x) \supset \psi_i = T$ for all i, $v(\psi) = T$, $y \overset{\psi_1,\tau}{\rightarrow} y_1$ ... $\overset{\psi_n,\tau}{\rightarrow} y_n$ $\overset{\psi,a}{\rightarrow} y'$ and xRy_i for all i and $x'Ry'$

iii. if xRy and $x \xrightarrow{\phi,a} \xi$, then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are propositions χ, $\psi, \psi_1, \ldots, \psi_n$ $(n \geq 0)$ and expressions $y_1, \ldots, y_n$ such that $\xi = \chi$, $s_\rho(x) \supset \psi_i = T$ for all i, $v(\psi) = T$, $y \xrightarrow{\psi_1,\tau} y_1 \ \ldots \ \xrightarrow{\psi_n,\tau} y_n \xrightarrow{\psi,a} \chi$ and xRy_i for all i

iv, v. like ii, iii with the roles of x and y interchanged.

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

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

vii. if $x \xrightarrow{\phi,a} \xi$, then for all valuations v such that $v(s_\rho(x) \wedge \phi) = T$, there are propositions ψ,χ such that $v(\psi) = T$, $y \xrightarrow{\psi,a} \chi$ and $\xi = \chi$

viii, ix. like vi, vii with the roles of x and y interchanged.

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.

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

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

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

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

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

i.e. $ACP^\tau ps$ is a sound and complete axiomatisation of the bisimulation model.

6. EXAMPLE: 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 (all specifications can be simply brought into a form that uses iteration rather than recursion). 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.

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

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

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

where $send_2(d) \mid read_2(d) = comm_2(d)$ (communication gives δ otherwise),

and $H = \{read_2(d), send_2(d) : d \in D\}$.

The system X obeys the usual specification of two coupled one-element buffers (as in [BAW90], page 106), after hiding all signals. 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

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

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

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

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

where communication is given by $send_2 \mid read_2 = comm_2$,

and encapsulation by $H = \{read_2, send_2\}$.

After abstracting from actions and signals at port 2, we obtain the same system as before.

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:

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

$$E^{ij}_d = (show_j(d) \wedge flag_i) \leftarrow (show_i(\varnothing) \wedge \neg flag_j):\rightarrow s_j \cdot E^{ij}.$$

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

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:

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

$$F^{ij}_d = (\neg ready_i \wedge show_j(d) \wedge \neg flag_j) \leftarrow (ready_j \wedge \neg flag_1):\rightarrow send_j \cdot G^{ij}_d.$$

$$G^{ij}_d = (\neg ready_i \wedge show_j(d) \wedge flag_j) \leftarrow \neg ready_j:\rightarrow reset_j \cdot F^{ij}_d.$$

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

where this is the free merge, i.e. this is a specification in PAps. Some calculations show that this system shows the required behaviour, after hiding all signals at port 2 and the additional signals introduced, and abstracting from the action set $I = \{read_2, send_2, comm_2, reset_2, reset_3\}$.

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.