Non Interleaving Process Algebra

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

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

We study a non interleaving subalgebra of a reduct of a model of ACP. The model discussed uses step bisimulation semantics. We can derive identities in this model with the help of the (interleaving) ACP calculus with multi-actions. We study the connection with Petri nets, and introduce causalities and a causal state operator.

1980 Mathematics Subject Classification (1985 revision): 68Q45, 68Q55, 68Q65, 68Q50.
1987 CR Categories: F.4.3, D.2.10, D.3.1, D.3.3.
Key words & Phrases: process algebra, interleaving, non-interleaving, true concurrency, Petri net, ACP.
Note: Partial support received by ESPRIT basic research action 7166, CONCUR2. The second author also received partial support from ESPRIT basic research action 6454, CONFER.

1. INTRODUCTION.

In this paper, we investigate a so-called true concurrency model from the viewpoint of ACP, more specifically a model with step bisimulation semantics. True concurrency models have in common that merge will not expand as a sum of products. We will call such models non interleaving. We obtain a non interleaving process algebra as a subalgebra of a reduct of an interleaving process algebra. Now the problem with developing the equational theory of this non interleaving process algebra is that the operator set of ACP is too much geared towards interleaving. This turns out to be the case in the simplest example of a non interleaving process algebra already, the case corresponding to step bisimulation semantics. This notion dates back at least to [NT84], see also [POM86] or [GV87] where it was called concurrent bisimulation.

In non interleaving process algebra causal connections between processes must be made explicit, as well as the absence of causal connections. Interleaving process algebra introduces causal connections via two mechanisms: sequential composition and (synchronous or asynchronous) communication. In fact, using the expansion theorem, all causal effects can be reduced to consequences of sequential composition. In the absence of an expansion theorem this reduction is not possible. Still, some reduction can be achieved, the communication mechanism can often be eliminated in favor of a more explicit handling of causalities. This is exactly what Petri nets [PET80] provide. In Petri nets, the tokens provide a bookkeeping of causalities which is so expressive that both sequential composition and communication cease to be necessary primitives.

We prove that the syntax of ACP is inadequate for eliminating communication in a very simple case: a buffer of capacity three in the setting of a non interleaving process algebra based on step bisimulation semantics. To remedy this lack of expressiveness of ACP we introduce two additional syntactic features:

- Petri elements. These are a special kind of atomic actions.
- a causal state operator.

First, we will present a version of ACP with explicit communication, where the mechanisms of parallel composition and communication are separated. Then we describe how to obtain a non interleaving algebra as the subalgebra of the serialisable processes of a reduct of an interleaving algebra. In this non interleaving model, we study the following issues:

- elimination of encapsulation and communication from a recursive specification of a buffer of capacity 2;
- impossibility of such an elimination for a buffer with capacity 3;
- we provide a translation of Petri net notation into the process algebra syntax, which is sound with respect to step bisimulation semantics
- we explain how to eliminate encapsulation and communication in favor of causalities and a causal state operator, for the case of a buffer of capacity 3.

Algebraic versions of Petri net descriptions have been investigated by many authors. We mention [BDH92], [GV87]. As related work we mention [KIE89], [CH89], [JPZ91].

It is not easy to compare these approaches and to explain the contribution of this paper in the context of the papers mentioned above. However, in comparison to all mentioned papers, we put more emphasis on equational reasoning. It seems to be the case that equational verification of (very simple) protocols is feasible in a non-interleaving semantics.

Of course, by restricting our attention to step bisimulation semantics, we make only a small step in the direction of causality based models. It remains to be seen whether extension of our approach to other non interleaving models is feasible.

This article is a slight revision of [BB93].

ACKNOWLEDGEMENT: Thanks to W. Reisig (TU München) for his helpful comments and suggestions.

2. PROCESS ALGEBRA WITH EXPLICIT COMMUNICATION.

We present a variant of the theory ACP of [BK84], [BW90] with explicit communication operator. As in ACP, parallel composition has three components, two left-merges and a synchronisation operator. In the last component, however, no communication is implemented, the synchronisation operator is free on atomic actions, i.e. for atoms a,b, $a \mid b$ represents a new atomic action (representing the synchronous execution of a and b). Communication is achieved by a separate operator, the communication operator, that takes the form of a renaming on atomic actions. This separation of concerns allows us to express the fact, that communication can be eliminated in certain cases.

$X + Y = Y + X$		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
$a \mid b \in A$		CCA
$a \mid b = b \mid a$		C1
$(a \mid b) \mid c = a \mid (b \mid c)$		C2
$\delta \mid a = \delta$		C3
$X \| Y = X \lfloor\!\lfloor Y + Y \lfloor\!\lfloor X + X \mid Y$		CM1
$a \lfloor\!\lfloor X = a\cdot X$		CM2
$a\cdot X \lfloor\!\lfloor Y = a\cdot(X \| Y)$		CM3
$(X + Y) \lfloor\!\lfloor Z = X \lfloor\!\lfloor Z + Y \lfloor\!\lfloor Z$		CM4
$a\cdot X \mid b = (a \mid b)\cdot X$		CM5
$a \mid b\cdot X = (a \mid b)\cdot X$		CM6
$a\cdot X \mid b\cdot Y = (a \mid b)\cdot(X \| Y)$		CM7
$(X+Y) \mid Z = X \mid Z + Y \mid Z$		CM8
$X \mid (Y+Z) = X \mid Y + X \mid Z$		CM9
$\partial_H(r) = r$	if $r \notin H$	D1
$\partial_H(r) = \delta$	if $r \in H$	D2
$\partial_H(\delta) = \delta$		DD
$\partial_H(a \mid b) = \partial_H(a) \mid \partial_H(b)$		DSM
$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$		D3
$\partial_H(X\cdot Y) = \partial_H(X)\cdot\partial_H(Y)$		D4
$\overline{\gamma}(r) = r$		CR1
$\overline{\gamma}(\delta) = \delta$		CR2
$\rho_{\overline{\gamma}}(a) = \overline{\gamma}(a)$		CR3
$\rho_{\overline{\gamma}}(X + Y) = \rho_{\overline{\gamma}}(X) + \rho_{\overline{\gamma}}(Y)$		CR4
$\rho_{\overline{\gamma}}(X\cdot Y) = \rho_{\overline{\gamma}}(X)\cdot\rho_{\overline{\gamma}}(Y)$		CR5

TABLE 1. $ACP_{ec}(CA,\overline{\gamma})$.

2.1 ACP_{ec}.

We describe a remodularisation of the process algebra ACP (Algebra of Communicating Processes) of [BK84], [BW90] with explicit communication operator. The axioms are in table 1, the signature of ACP_{ec} is as follows:

Sorts:

P	sort of processes
$A \subseteq P$	subsort of atomic (multi-)actions.

Constants:

$\delta \in A$	inaction (deadlock).
$r \in A$	core atomic action, for each $r \in CA$

Functions:

$+: P \times P \to P$	alternative composition, sum.
$\cdot: P \times P \to P$	sequential composition, product.
$\|: P \times P \to P$	parallel composition, merge.
$⌊⌊: P \times P \to P$	left-merge.
$\mid: P \times P \to P$	synchronisation merge.
$\partial_H: P \to P$	encapsulation operator, for each $H \subseteq CA$
$\rho_{\overline{\gamma}}: P \to P$	communication operator

Variables:

$X,Y,Z,... \in P$	process variables
$a,b,c,... \in A$	(multi-)action variables.

The axioms in table 1 have the form of a first order theory. For instance, axiom C1 should be read as

$$\forall a \in A\, \forall b \in A\, (a \mid b = b \mid a).$$

Axioms D1, D2 are actually axiom schemes: there is such an axiom for each core atom r. The theory has two parameters: first, a finite set CA of core atomic actions. The set of atomic actions A is generated from CA by application of the synchronisation merge $\mid$. Elements of the set A are called atomic actions, multi-actions (using the terminology of [BB91]) or just actions. The inaction constant δ is in A, but not in CA, is not a core action.

The synchronisation merge is free over atoms, just restricted by axioms C1-3. Thus, for atomic actions a,b, $a \mid b$ represents a new atomic action. It should be noted that this construction yields infinitely many atomic actions. CCA (Communication Closure Axiom) simply expresses that the synchronisation of two (multi-)actions is again a (multi-)action. In this equational specification, all elements of A are either equal to δ or can be written in the form:

$$r_1 \mid r_2 \mid ... \mid r_n \qquad \text{with } n > 0,\ r_i \in CA.$$

The second parameter is the communication function $\overline{\gamma}$, a renaming on atoms that leaves core atoms and δ fixed. The connection with [BK84] is as follows: there, ACP is parametrised by a communication function $\gamma: CA \times CA \to CA \cup \{\delta\}$. Given such a binary function γ, we can define $\overline{\gamma}$ on A as follows:

$$\overline{\gamma}(r) = r \qquad \overline{\gamma}(\delta) = \delta \qquad \overline{\gamma}(a \mid b) = \gamma(\overline{\gamma}(a), \overline{\gamma}(b))$$

If we write $\|_\gamma$ for the merge of [BK84] with communication function γ then the following identity connects both approaches:

$$X \|_\gamma Y = \rho_{\overline{\gamma}}(X \| Y).$$

2.2 READ-SEND COMMUNICATION.

We give a specific instantiation of the set of core atoms CA and communication on CA. We assume the set of messages is a given non-empty finite set N, and we assume given a non-empty finite set of communication ports $\mathcal{M}$. For each $i \in N$ and $m \in \mathcal{M}$ we have the following core atoms:

- $r_m(i)$ read message i at port m,
- $s_m(i)$ send message i at port m,
- $c_m(i)$ communicate message i at port m.

This notation was introduced in [BK86]. Thus, we have $CA = \{r_m(i), s_m(i), c_m(i) \mid i \in N, m \in \mathcal{M}\}$. On this action set, communication is defined by means of the axioms in table 2 ($a \in A$) in addition to axioms CR1,2. These axioms are actually axiom schemes, so for instance the first axiom exists for every $i \in N$ and $m \in \mathcal{M}$. Notice that the condition $a \neq \delta$ is equivalent to $ports(a) \neq \emptyset$. We call this particular instance of $ACP_{ec}(CA,\overline{\gamma})$ $ACP_{ec}(rsc(\mathcal{M},N))$

$\overline{\gamma}(r_m(i) \mid s_m(i)) = c_m(i)$	
$\overline{\gamma}(r_m(i) \mid s_m(j)) = \delta$	if $i \neq j$
$\overline{\gamma}(r_m(i) \mid s_m(j) \mid a) = \delta$	if $i \neq j$
$\overline{\gamma}(r_m(i) \mid r_m(j)) = \delta$	
$\overline{\gamma}(r_m(i) \mid r_m(j) \mid a) = \delta$	
$\overline{\gamma}(s_m(i) \mid s_m(j)) = \delta$	
$\overline{\gamma}(s_m(i) \mid s_m(j) \mid a) = \delta$	
$\overline{\gamma}(c_m(i) \mid a) = \delta$	if $m \in ports(a)$
$\overline{\gamma}(a \mid b) = \overline{\gamma}(a) \mid \overline{\gamma}(b)$	if $ports(a) \cap ports(b) = \emptyset$
$ports(r_m(i)) = ports(s_m(i)) = ports(c_m(i)) = \{m\}$	
$ports(\delta) = \emptyset$	
$ports(a \mid b) = ports(a) \cup ports(b)$	if $a \neq \delta, b \neq \delta$

TABLE 2. Read-send communication

As an example, we calculate

$$r_1(i) \| s_1(i) = r_1(i)\cdot s_1(i) + s_1(i)\cdot r_1(i) + r_1(i) \mid s_1(i)$$

$$\rho_{\overline{\gamma}}(r_1(i) \| s_1(i)) = r_1(i)\cdot s_1(i) + s_1(i)\cdot r_1(i) + c_1(i)$$

$$\partial_H \circ \rho_{\overline{\gamma}}(r_1(i) \| s_1(i)) = c_1(i), \text{ if } H = \{r_1(i), s_1(i) : i \in N\}.$$

2.3 BUFFERS.

We can specify a one-item buffer with input port m and output port k as follows:

$$B^{mk} = \sum_{i \in N} r_m(i) \cdot s_k(i) \;\cdot B^{mk}.$$

Put $H(m) = \{s_m(i), r_m(i) : i \in N\}$ for $m \in \mathcal{M}$, using the read-send communication function $\overline{\gamma}$, then we can define a two-item buffer and a three-item buffer as follows:

$$C = \partial_{H(2)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel B^{23})$$

$$D = \partial_{H(2) \cup H(3)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel B^{23} \parallel B^{34})$$

In case the set of messages N is a singleton, $|N| = 1$, we can leave out the data elements and simply write e.g.: $B^{12} = r_1 \cdot s_2 \cdot B^{12}$.

In this case, using the axioms of ACP_{ec}, we can derive the following recursive specification for the two-item buffer (where $C = \partial_{H(2)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel B^{23})$ and $C' = \partial_{H(2)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel s_3 \cdot B^{23})$):

$$C = r_1 \cdot c_2 \cdot C'.$$

$$C' = (r_1 \parallel s_3) \cdot c_2 \cdot C'.$$

Similarly, for the three-item buffer we can derive the following specification:

$$D = D_{000} = r_1 \cdot D_{100}$$

$$D_{100} = c_2 \cdot D_{010}$$

$$D_{010} = r_1 \cdot D_{110} + c_3 \cdot D_{001} + (r_1 \mid c_3) \cdot D_{101}$$

$$D_{110} = c_3 \cdot D_{101}$$

$$D_{001} = r_1 \cdot D_{101} + s_4 \cdot D_{000} + (r_1 \mid s_4) \cdot D_{100}$$

$$D_{101} = c_2 \cdot D_{011} + s_4 \cdot D_{100} + (c_2 \mid s_4) \cdot D_{010}$$

$$D_{011} = r_1 \cdot D_{111} + s_4 \cdot D_{010} + (r_1 \mid s_4) \cdot D_{110}$$

$$D_{111} = s_4 \cdot D_{110}.$$

We show the transition system of D in fig. 1.

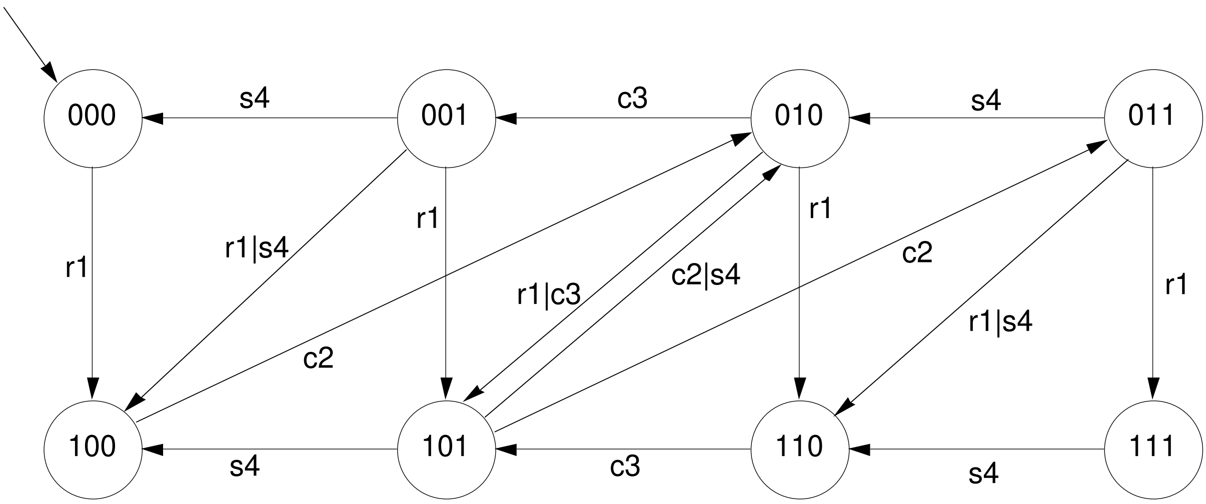

FIGURE 1.

2.4 OPERATIONAL SEMANTICS.

We present a model for a theory ACP_{ec} using structured operational semantics. This is fairly standard. The rules below are adapted from [BW90].

In table 3, $a,b \in A - \{\delta\}$, X,Y,Z are processes. In the encapsulation rules, we use the function $core(a)$. If a is of the form $r_1 \mid r_2 \mid \ldots \mid r_n$ with $n > 0$, $r_i \in CA$, then $core(a) = \{r_1, \ldots, r_n\}$.

$$a \xrightarrow{a} \surd$$

$$\frac{X \xrightarrow{a} X'}{X+Y \xrightarrow{a} X',\ Y+X \xrightarrow{a} X'} \qquad \frac{X \xrightarrow{a} \surd}{X+Y \xrightarrow{a} \surd,\ Y+X \xrightarrow{a} \surd}$$

$$\frac{X \xrightarrow{a} X'}{X \cdot Y \xrightarrow{a} X' \cdot Y} \qquad \frac{X \xrightarrow{a} \surd}{X \cdot Y \xrightarrow{a} Y}$$

$$\frac{X \xrightarrow{a} X'}{X \| Y \xrightarrow{a} X' \| Y,\ Y \| X \xrightarrow{a} Y \| X',\ X \mathbin{|\!\!\lfloor} Y \xrightarrow{a} X' \| Y}$$

$$\frac{X \xrightarrow{a} \surd}{X \| Y \xrightarrow{a} Y,\ Y \| X \xrightarrow{a} Y,\ X \mathbin{|\!\!\lfloor} Y \xrightarrow{a} Y}$$

$$\frac{X \xrightarrow{a} X',\ Y \xrightarrow{b} Y'}{X \| Y \xrightarrow{a \mid b} X' \| Y',\ X \mid Y \xrightarrow{a \mid b} X' \| Y'}$$

$$\frac{X \xrightarrow{a} \surd,\ Y \xrightarrow{b} \surd}{X \| Y \xrightarrow{a \mid b} \surd,\ X \mid Y \xrightarrow{a \mid b} \surd}$$

$$\frac{X \xrightarrow{a} X',\ Y \xrightarrow{b} \surd}{X \| Y \xrightarrow{a \mid b} X',\ X \mid Y \xrightarrow{a \mid b} X',\ Y \| X \xrightarrow{b \mid a} X',\ Y \mid X \xrightarrow{b \mid a} X'}$$

$$\frac{X \xrightarrow{a} X',\ H \cap core(a) = \emptyset}{\partial_H(X) \xrightarrow{a} \partial_H(X')} \qquad \frac{X \xrightarrow{a} \surd,\ H \cap core(a) = \emptyset}{\partial_H(X) \xrightarrow{a} \surd}$$

$$\frac{X \xrightarrow{a} X',\ \bar\gamma(a) \neq \delta}{\rho_{\bar\gamma}(X) \xrightarrow{\bar\gamma(a)} \rho_{\bar\gamma}(X')} \qquad \frac{X \xrightarrow{a} \surd,\ \bar\gamma(a) \neq \delta}{\rho_{\bar\gamma}(X) \xrightarrow{\bar\gamma(a)} \surd}$$

TABLE 3. Operational semantics for $ACP_{ec}(CA,\bar\gamma)$.

In this paper, attention will be restricted to regular processes only. Therefore, we give a structured operational semantics for regular processes, and so we only need to look at finite linear recursive specifications. If E is such a system of linear equations over variables $X_1, ..., X_n$, then we add constants $\langle X_i \mid E\rangle$ for each $i = 1,...,n$. If $X_i = a_1 \cdot X_{i_1} + ... + a_m \cdot X_{i_m} + b_1 + ... + b_p$ is an equation of E, such that $a_j \neq \delta$, $b_k \neq \delta$, then we add rules

$\langle X_i \mid E\rangle \xrightarrow{a_j} \langle X_{i_j} \mid E\rangle$ $\qquad$ $\langle X_i \mid E\rangle \xrightarrow{b_k} \surd$ $\qquad$ for $j = 1,...,m$ and $k = 1,...,p$.

On the set of transition systems obtained by these rules, we define (strong) bisimulation in the usual way. This gives us a model $\mathcal{G} = \mathcal{G}(CA, \overline{\gamma})$ of transition systems modulo bisimulation. We can also obtain this model by considering the set of finite process graphs, and define all operators on this model, as in [BW90]. As a set of atomic actions can be performed simultaneously (as a multi-action) this model corresponds to step bisimulation semantics. This notion dates back at least to [NT84], see also [POM86] or [GV87] where it was called concurrent bisimulation.

2.5 CAUSE ADDITION.

Our aim is now to give an algebraic specification of Petri nets. It is already known how to give a Petri net semantics of process algebra, see e.g. [GV87]. There, the constants and operators of ACP are defined on safe labelled marked nets with bounded parallelism and non-empty initial marking. This suffices to find a Petri net representing a closed term. It is easy to also define linear recursion on such Petri nets. For an approach that can handle more than just linear recursion, see [GOL88].

Here, we want to go the other direction.

Let C be a given set, the set of *causes*. We define two operators, for every $c \in C$:

$^c\_ : P \to P$ $\qquad$ input cause addition

$\_^c : P \to P$ $\qquad$ output cause addition.

We also define the causes of a process by means of an operator

$\text{causes}: P \to 2^C$ $\qquad$ set of causes of a process.

We have the axioms in table 4 ($a,b \in A$, $c,d \in C$).

These axioms allow us to write $^Va^W$ for $^{c_1}(...^{c_n}((.(a^{d_1})...^{d_m})$, for multisets of causes $V = \{c_1,...,c_n\}$, $W = \{d_1,...,d_m\}$. We can write all atomic actions in the form $^Va^W$, where $\text{causes}(a) = \varnothing$, by allowing $V = \varnothing$ or $W = \varnothing$, with the convention that $^\varnothing a = a^\varnothing = a$. We call the atomic actions $^Va^W$ with $\text{causes}(a) = \varnothing$ *Petri elements*.

We can also define the set of causes of a recursively defined process by means of the equation:

$$\text{causes}(X) = \cup_{n\geq 1}\, \text{causes}(\pi_n(X)).$$

Here, we use the projection operators π_n, see [BW90].

We need some notation for multisets. If V is a multiset over C, then $\#c(V)$ is the multiplicity of $c \in C$ in V (the number of times c appears in V; $\text{set}(V)$ is the set of V, i.e. $c \in \text{set}(V)$ iff $\#c(V) \geq 1$. If T is a set over C, and V a multiset over C, then $V\upharpoonright T$ is V restricted to T, i.e. $\#c(V\upharpoonright T) = \#c(V)$ if $c \in T$ and $\#c(V\upharpoonright T) = 0$ if $c \notin T$. The set operators $\cup$, -, $\subseteq$ are also used for multisets, i.e. $V \subseteq W$ if $\#c(V) \leq \#c(W)$ for all $c \in C$, $\#c(V\cup W) = \#c(V) + \#c(W)$, $\#c(V-W) = \max(0, \#c(V) - \#c(W))$ for all $c \in C$.

${}^{c}a \in A$	$a^{c} \in A$	
${}^{c}\delta = \delta^{c} = \delta$		
${}^{c}a = {}^{c}b \Rightarrow a = b$	$a^{c} = b^{c} \Rightarrow a = b$	
${}^{c}(a^{d}) = ({}^{c}a)^{d}$		
${}^{c}({}^{d}a) = {}^{d}({}^{c}a)$	$(a^{c})^{d} = (a^{d})^{c}$	
$({}^{c}a) \mid b = {}^{c}(a \mid b)$	$(a^{c}) \mid b = (a \mid b)^{c}$	
${}^{c}(X + Y) = {}^{c}X + {}^{c}Y$	$(X + Y)^{c} = X^{c} + Y^{c}$	
${}^{c}(X \cdot Y) = {}^{c}X \cdot Y$	$(X \cdot Y)^{c} = X \cdot Y^{c}$	
$causes(r) = \varnothing$	for each $r \in CA$	
$causes(\delta) = \varnothing$		
$causes({}^{c}a) = causes(a^{c}) = \{c\} \cup causes(a)$		if $a \neq \delta$
$causes(a \mid b) = causes(a) \cup causes(b)$		if $a \neq \delta$, $b \neq \delta$
$causes(X + Y) = causes(X) \cup causes(Y)$		
$causes(a \cdot X) = causes(a) \cup causes(X)$		if $a \neq \delta$

TABLE 4. Cause addition and causes of a process.

2.6 CAUSAL STATE OPERATOR.

Now we add a special kind of state operator. State operators were introduced in [BB88]. A state operator has a certain state space, and functions act and eff modifying states and actions. Here we have a specific state space (a pair of a set and a multiset over C) and specific act and eff functions (implicitly given in table 5).

$\lambda^{C}_{T,V}: P \to P$ causal state operator. T a set over C, V a multiset over C with $set(V) \subseteq T$.

T contains the causes that are encapsulated by the causal state operator, V contains the initial multiplicities of each of these causes. This is inspired by, but not quite the same as the causality mechanism of [BKT85].

Axioms are shown in table 5. Here, T is a set over C, V,W,Z are multisets over C.

$\lambda^{C}_{T,V}(\delta) = \delta$	
$\lambda^{C}_{T,V}({}^{W}a^{Z}) = {}^{W\restriction(C-T)}a^{Z\restriction(C-T)}$	if $W\restriction T \subseteq V$, $causes(a)=\varnothing$, $a\neq\delta$
$\lambda^{C}_{T,V}({}^{W}a^{Z}) = \delta$	if $W\restriction T - V \neq \varnothing$
$\lambda^{C}_{T,V}({}^{W}a^{Z} \cdot X) = \lambda^{C}_{T,V}({}^{W}a^{Z}) \cdot \lambda^{C}_{T,((V\cup Z)-W)\restriction T}(X)$	if $causes(a)=\varnothing$
$\lambda^{C}_{T,V}(X + Y) = \lambda^{C}_{T,V}(X) + \lambda^{C}_{T,V}(Y)$	

TABLE 5. Causal state operator.

Notice that we have $causes(\lambda^{C}_{T,V}(X)) \subseteq causes(X)\restriction(C-T)$. Notice that these equations allow us to eliminate the causal state operator from all closed terms.

We call the resulting process algebra specification, ACP_{ec} plus cause addition and causal state operator, $ACP_{pe}(CA, \overline{\gamma}, C)$. Operational rules for this operator are straightforward, see table 6. This gives us a operational model $\mathcal{G}(CA, \overline{\gamma}, C)$.

$$\frac{X \xrightarrow{{}^{W}a^{Z}} X',\ causes(a)=\varnothing,\ W\restriction T \subseteq V}{\lambda^{C}_{T,V}(X) \xrightarrow{{}^{W\restriction(C-T)}a^{Z\restriction(C-T)}} \lambda^{C}_{T,((V\cup Z)-W)\restriction T}(X')}$$

$$\frac{X \xrightarrow{{}^{W}a^{Z}} \surd,\ causes(a)=\varnothing,\ W\restriction T \subseteq V}{\lambda^{C}_{T,V}(X) \xrightarrow{{}^{W\restriction(C-T)}a^{Z\restriction(C-T)}} \surd}$$

TABLE 6. Operational semantics for causal state operator.

2.7 THEOREM: $k \notin causes(X) \cup causes(Y) \;\Rightarrow\; X \cdot Y = \lambda^{C}_{\{k\},\varnothing}(X^{k} \parallel {}^{k}Y)$.

PROOF: This theorem can be proven for all closed terms over ACP_{pe} by means of structural induction, and is also valid in the model. We conjecture, however, that it cannot be derived from the theory as an open conditional equation.

This identity substantiates Milner's point of view (see e.g. [MIL93]) that sequential composition is not necessarily a primitive operator. We agree that sequential composition is probably not a primitive computational concept, but, as it can be expressed in many ways using various combinations of other primitives, it seems to be well placed as a primitive for a mathematical theory of processes.

2.8 NORMAL FORM THEOREM. Let the set of causes C be infinite. Then every finite closed process expression over the signature $CA,\delta,+,\cdot,\parallel,\mathbin{⫗},{}^{c}\_,\_^{c}$ can be written in the form $\lambda^{C}_{T,V}(r_1 \parallel ... \parallel r_n)$, where each r_i is a finite sum of Petri elements.
PROOF: Omitted.

2.9 PETRI NETS.
We can use the algebra ACP_{pe} to specify Petri nets. A transition t in a Petri net with label $r \in CA$ and with multiset of input places V and multiset of output places W can be modeled by a Petri element ${}^{V}r^{W}$. In this case, we put $[t] = {}^{V}r^{W}$.

If we have a completely unfolded (i.e. no cycles) finite Petri net P with set of places T, set of transitions $\{t_1, ..., t_n\}$ and initial marking V (a multi-set of places). The algebraic interpretation of the t_i, $[t_i]$, is given above. Now the semantics of P, $[P]$, is given by:

$$[P] = \lambda^{T}_{T,V}([t_1] \parallel [t_2] \parallel ... \parallel [t_n]).$$

2.10 EXAMPLE.
Consider the term $r\cdot(s \parallel t)\cdot u)$ $(r,s,t,u \in CA)$. Using the semantics of [GV87], we obtain the Petri net given in fig. 2.

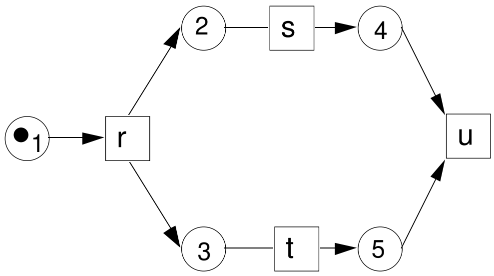

FIGURE 2.

Label the places $C = \{1,2,3,4,5\}$ as shown in fig. 2. By the semantics in 2.9, we obtain the term

$$\lambda^{C}_{C,\{1\}}({}^{\{1\}}r^{\{2,3\}} \parallel {}^{\{2\}}s^{\{4\}} \parallel {}^{\{3\}}t^{\{5\}} \parallel {}^{\{4,5\}}u).$$

We usually leave out the curly brackets in input and output causes and write

$$\lambda^{C}_{C,\{1\}}({}^{1}r^{2,3} \parallel {}^{2}s^{4} \parallel {}^{3}t^{5} \parallel {}^{4,5}u).$$

Using the axioms in table 5 and the expansion theorem of ACP_{pe}, we obtain

$$\lambda^{C}_{C,\{1\}}({}^{1}r^{2,3} \parallel {}^{2}s^{4} \parallel {}^{3}t^{5} \parallel {}^{4,5}u) = r \cdot \lambda^{C}_{C,\{2,3\}}({}^{2}s^{4} \parallel {}^{3}t^{5} \parallel {}^{4,5}u) =$$

$$= r \cdot (s \cdot \lambda^{C}_{C,\{3,4\}}({}^{3}t^{5} \parallel {}^{4,5}u) + t \cdot \lambda^{C}_{C,\{2,4\}}({}^{2}s^{4} \parallel {}^{4,5}u) + (s \mid t) \cdot \lambda^{C}_{C,\{4,5\}}({}^{4,5}u)) =$$

$$= r \cdot (s \cdot t \cdot \lambda^{C}_{C,\{4,5\}}({}^{4,5}u) + t \cdot s \cdot \lambda^{C}_{C,\{4,5\}}({}^{4,5}u) + (s \mid t) \cdot \lambda^{C}_{C,\{4,5\}}({}^{4,5}u)) =$$

$$= r \cdot (s \cdot t + t \cdot s + (s \mid t)) \cdot \lambda^{C}_{C,\{4,5\}}({}^{4,5}u) = r \cdot (s \parallel t) \cdot u.$$

We see that we obtain the same term we started from.

2.11 PETRI NETS WITH LOOPS.

In some cases, it is profitable to work with Petri nets involving loops. In that case, transitions may fire more than once. In this case, we give the algebraic interpretation of a transition t with label r, input causes V and output causes W by means of the following recursion equation:

$$[t] = {}^{V}r^{W} \cdot [t].$$

2.12 EXAMPLE.

Consider the two-item buffer of 2.3: $C = \partial_{H(2)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel B^{23})$. The corresponding Petri net is given in fig. 3.

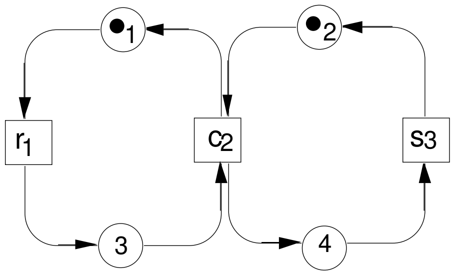

FIGURE 3.

Use the set of place labels $L = \{1,2,3,4\}$ and the set of transition equations:

$T_1 = {}^{1}r_1{}^{3} \cdot T_1$ $\qquad$ $T_2 = {}^{2,3}c_2{}^{1,4} \cdot T_2$ $\qquad$ $T_3 = {}^{4}s_3{}^{2} \cdot T_3$.

Put $C = \lambda^L_{L,\{1,2\}}(T_1 \parallel T_2 \parallel T_3)$, $C' = \lambda^L_{L,\{1,4\}}(T_1 \parallel T_2 \parallel T_3)$.

Using the axioms in table 6 and the expansion theorem of ACP_{pe}, we obtain

$C = \lambda^L_{L,\{1,2\}}(T_1 \parallel T_2 \parallel T_3) = r_1 \cdot \lambda^L_{L,\{2,3\}}(T_1 \parallel T_2 \parallel T_3) = r_1 \cdot c_2 \cdot \lambda^L_{L,\{1,4\}}(T_1 \parallel T_2 \parallel T_3) =$

$= r_1 \cdot c_2 \cdot C'$

$C' = \lambda^L_{L,\{1,4\}}(T_1 \parallel T_2 \parallel T_3) =$

$= r_1 \cdot \lambda^L_{L,\{2,4\}}(T_1 \parallel T_2 \parallel T_3)) + s_3 \cdot \lambda^L_{L,\{1,2\}}(T_1 \parallel T_2 \parallel T_3) + (r_1 \mid s_3) \cdot \lambda^L_{L,\{2,3\}}(T_1 \parallel T_2 \parallel T_3) =$

$= (r_1 \cdot s_3 + s_3 \cdot r_1 + r_1 \mid s_3) \cdot \lambda^L_{L,\{2,3\}}(T_1 \parallel T_2 \parallel T_3) = (r_1 \parallel s_3) \cdot c_2 \cdot C'$.

We see that we get back the same specification we obtained in 2.3.

2.13 THREE-ITEM BUFFER.

Consider the three-item buffer of 2.3: $D = \partial_{H(2) \cup H(3)} \circ \rho_{\overline{\gamma}}(B^{12} \parallel B^{23} \parallel B^{34})$. The corresponding Petri net is given in fig. 4.

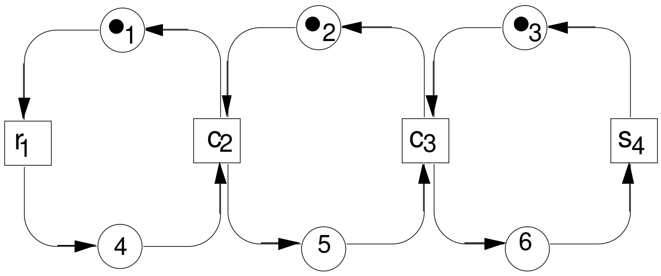

FIGURE 4.

Use the set of place labels $L = \{1,2,3,4,5,6\}$ and the set of transition equations:

$T_1 = {}^1r_1{}^4 \cdot T_1$ $\quad T_2 = {}^{2,4}c_2{}^{1,5} \cdot T_2$ $\quad T_3 = {}^{3,5}s_3{}^{2,6} \cdot T_3$. $\quad T_4 = {}^6s_4{}^3 \cdot T_4$.

Put $D = D_{000} = \lambda^L_{L,\{1,2,3\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$ $\quad D_{001} = \lambda^L_{L,\{1,2,6\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$

$D_{100} = \lambda^L_{L,\{2,3,4\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$ $\quad D_{101} = \lambda^L_{L,\{2,4,6\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$

$D_{010} = \lambda^L_{L,\{1,3,5\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$ $\quad D_{011} = \lambda^L_{L,\{1,5,6\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$

$D_{110} = \lambda^L_{L,\{3,4,5\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$ $\quad D_{111} = \lambda^L_{L,\{4,5,6\}}(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$.

It is now easy to obtain the specification of 2.3.

2.14 THREE-ITEM BUFFER: SECOND SPECIFICATION.

In 2.13, we obtained a specification for the three-item buffer that does not use encapsulation or communication, but uses 6 causes. We can do this also with less causes, as the following specification shows. This specification is obtained by "internalising" causes 1,3,4,6 of the specification in 2.13, they are replaced by a sequential composition. Put $L = \{2,5\}$. Consider:

$Z = \lambda^L_{L,\{2\}}(X \parallel Y)$ $\quad X = r_1 \cdot {}^2c_2{}^5 \cdot X$ $\quad Y = {}^5c_3{}^2 \cdot s_4 \cdot Y$.

Then we obtain $Z = D$ by using the specification in 2.13 and the following identifications:

$D_{000} = \lambda^L_{L,\{2\}}(X \parallel Y)$ $\quad D_{001} = \lambda^L_{L,\{2\}}(X \parallel s_4 \cdot Y)$

$D_{100} = \lambda^{L}_{L,\{2\}}({}^2c_2{}^5 \cdot X \parallel Y)$ $D_{101} = \lambda^{L}_{L,\{2\}}({}^2c_2{}^5 \cdot X \parallel s_4 \cdot Y)$

$D_{010} = \lambda^{L}_{L,\{5\}}(X \parallel Y)$ $D_{011} = \lambda^{L}_{L,\{5\}}(X \parallel s_4 \cdot Y)$

$D_{110} = \lambda^{L}_{L,\{5\}}({}^2c_2{}^5 \cdot X \parallel Y)$ $D_{111} = \lambda^{L}_{L,\{5\}}({}^2c_2{}^5 \cdot X \parallel s_4 \cdot Y)$.

2.15 THREE-ITEM BUFFER: THIRD SPECIFICATION.

We have the following interesting specification of the three-item buffer. In this specification, we use the left-merge operator, not to obtain interleaving, but only to give a guarded specification of what is in fact a merge of an unbounded set of processes. Put $L = \{0,1\}$.

$D = \lambda^{L}_{\{0\},\{0,0,0\}}(K_0)$

$K_0 = \lambda^{L}_{\{1\},\varnothing}({}^0r_1{}^1 \cdot {}^0c_2{}^1 \cdot {}^0c_3{}^1 \cdot s_4{}^1 \mathbin{⫡} K_1)$ $K_1 = \lambda^{L}_{\{0\},\varnothing}({}^1r_1{}^0 \cdot {}^1c_2{}^0 \cdot {}^1c_3{}^0 \cdot s_4{}^0 \mathbin{⫡} K_0)$.

3. A NON INTERLEAVING PROCESS ALGEBRA.

We obtain a non interleaving process algebra as a subalgebra of the model $\mathcal{G}(CA, \overline{\gamma}, C)$. This subalgebra consists of all serialisable processes. This notion will be defined first.

3.1 SERIALISABLE PROCESSES.

Let p be a process in $\mathcal{G}(CA, \overline{\gamma}, C)$. We say that p is *serialisable* if for all states s of p,

- whenever $s \overset{V a \mid b W}{\rightarrow} \surd$ with $causes(a \mid b) = \varnothing$, then there are multi-sets of causes V', V'', W', W'' with $V' \cup V'' = V$ and $W' \cup W'' = W$ and a state s' such that $s \overset{V'aW'}{\rightarrow} s' \overset{V''bW''}{\rightarrow} \surd$
- whenever $s \overset{V a \mid b W}{\rightarrow} s''$ with $causes(a \mid b) = \varnothing$, then there are multi-sets of causes V', V'', W', W'' with $V' \cup V'' = V$ and $W' \cup W'' = W$ and a state s' such that $s \overset{V'aW'}{\rightarrow} s' \overset{V''bW''}{\rightarrow} s''$.

Let $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ be the set of all serialisable processes from $\mathcal{G}(CA, \overline{\gamma}, C)$.

3.2 THEOREM: $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ is closed under the operators $+, \cdot, \parallel, \mathbin{⫡}, \partial_H, \rho_{\overline{\gamma}}, \lambda^{C}_{T,V}, {}^c\_, \_{}^c$ and linear recursion.

PROOF: Omitted.

3.3 REMARKS.

1. Thus, $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ is closed under all operators of ACP_{pe} except for the synchronisation merge $\mid$ (as the process $r \mid s$, for core atoms r,s, is obviously not serialisable), and so $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ is a subalgebra of $\mathcal{G}(CA, \overline{\gamma}, C)$ for the restricted signature not involving $\mid$.
2. $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ is a non interleaving process algebra, as the process $r \parallel s$, for core atoms r,s, is not expressible with atomic actions, + and $\cdot$. Notice that atoms are elements of A, but atoms can also be characterised as those process that cannot be split, either as a sum or as a sequential composition, in a nontrivial way.
3. In the algebra $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ the objective of elimination (expansion of merge) disappears. Instead, it becomes an objective to remove the communication operator, and to replace it by the weaker causal

state operator that originates from Petri net theory. Thus, we eliminate communication in favor of causality.
4. The definitions in 2.12-15 are all given in the signature of 3.2. Equality of the defined processes is shown using equational reasoning in the interleaving extension algebra $\mathcal{G}(\mathrm{CA}, \overline{\gamma}, \mathrm{C})$.

3.4 TWO-ITEM BUFFER.
The two-item buffer of 2.3 satisfies in $\mathcal{G}_{\mathrm{ser}}(\mathrm{CA}, \overline{\gamma}, \mathrm{C})$ the specification

$$C = r_1 \cdot c_2 \cdot C'.$$
$$C' = (r_1 \parallel s_3) \cdot c_2 \cdot C'.$$

This follows immediately from the definition of the model and the fact that this specification is written in the reduced signature.

Notice that the specification without the state operator for the three-item buffer in 2.3 is not in the reduced signature. In fact, we have the following theorem.

3.5 THEOREM: Let D be the three-item buffer defined in 2.3. There is no service specification for this process in $\mathcal{G}_{\mathrm{ser}}(\mathrm{CA}, \overline{\gamma}, \mathrm{C})$, that does not use the communication operator or the causal state operator.

PROOF: Suppose $E(X_1, ..., X_n)$ specifies D and does not use the communication operator or the causal state operator. First of all, if $\parallel$ does not occur in E, then D cannot perform any steps involving a true multi-action, and this is a contradiction, for e.g. $D \xrightarrow{r1} \xrightarrow{c2} \xrightarrow{r1 \mid c3}$.

So, suppose $\parallel$ occurs in the specification. Then there must be an outermost merge. It follows that D must have a state T of the form

(i) $P \parallel Q + R$, or
(ii) $(P \parallel Q) \cdot S + R$.

First of all, we observe that both P and Q must be finite. To see this, assume e.g. that P is infinite and let σ be an infinite trace of P and let a be some initial core atomic action of Q. Everywhere along σ, a is enabled. Inspection of the state graph of D (fig. 1 in 2.3) brings about that no infinite path exists along which some action is continuously enabled. In fact, every infinite trace must contain all actions (perhaps as part of a multi-action) and just after an action has been performed, it is not enabled.

As both P and Q are finite, from T a trace can be found that leads to

(i) $a \parallel b + R$, or
(ii) $(a \parallel b) \cdot S + R$

for appropriate $a,b \in \mathrm{CA}$. As D is perpetual, (i) is in fact impossible so we are left with (ii). First notice that $a \neq b$. Otherwise a trace $a \cdot a \cdot a \cdot ...$ would exist from T which is impossible by inspection of fig. 1. We conclude that the transition system for D must contain as a substructure one of the two forms in fig. 5 below. By inspection, one observes that no T can have this property.

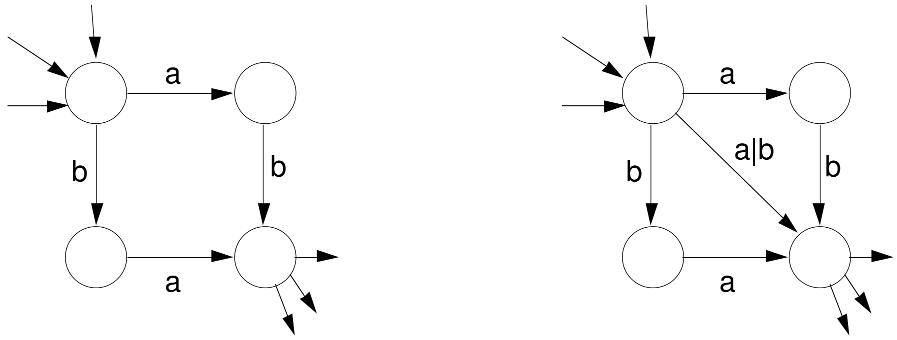

FIGURE 5.

3.6 DEFINITION.

We define two subalgebras of $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$.

1. $\mathcal{G}_1(CA)$ is the minimal subalgebra of the reduct of $\mathcal{G}_{ser}(CA, id, \varnothing)$ to the signature $CA, +, \cdot, \parallel, \mathbin{|\!\!\lfloor}$. This algebra also uses step bisimulation semantics. It is obvious that $CA, +, \cdot$ is not a set of generators, as $r \parallel s$ cannot be expressed using this set.

2. $\mathcal{G}_2(CA, \overline{\gamma}, C)$ is the minimal subalgebra of the reduct of $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ to the signature of ACP_{ec} plus cause addition.

3.7 THEOREM. $\mathcal{G}_2(CA, \overline{\gamma}, C)$ is not generated by the signature $CA, \delta, +, \cdot, \parallel, \mathbin{|\!\!\lfloor}, {}^c\_, \_^c$.

PROOF: Consider the process $P = \lambda^{\{1\}}_{\{1\},\varnothing}(a^1 \cdot b \parallel c \cdot^1 d)$ with $a,b,c,d \in CA$. This process is in $\mathcal{G}_2(CA, \overline{\gamma}, C)$, as we can prove $P = a \cdot (b \parallel c \cdot d) + c \cdot a \cdot (b \parallel d) + (a \mid c) \cdot b \cdot d$. We claim that this process cannot be written using just $CA, +, \cdot, \parallel, \mathbin{|\!\!\lfloor}$. To see this, notice that every execution of P involves exactly 4 core actions. If $P = Q \parallel R$, then every execution of Q involves the same number of core actions (1,2 or 3), and every execution of R involves the same number of core actions (3,2 or 1). Further, each core action occurs either in Q or in R but not in both. Also, Q and R cannot involve $\parallel$ because otherwise we would obtain a synchronisation of three core actions. If Q or R involve $\mathbin{|\!\!\lfloor}$ then the left hand argument must be a sum of core actions because otherwise we again obtain a ternary synchronisation. But in that case $\mathbin{|\!\!\lfloor}$ can be expanded with + and $\cdot$.

Thus, we conclude that Q,R are terms over $CA, +, \cdot, {}^c\_, \_^c$. At this point, finitely many options are left, and it can be checked in a systematic way that none of them works.

3.8 SOME OPEN QUESTIONS.

i. Is it possible to find finite equational axiomatisations of $\mathcal{G}_1(CA)$, $\mathcal{G}_2(CA, \overline{\gamma}, C)$. If not, do such axiomatisations exist by means of less auxiliary objects and functions than are used in the present construction of these algebras.

ii. Which collections of operators generate $\mathcal{G}_2(CA, \overline{\gamma}, C)$. For example, is $CA, +, \cdot, \parallel, \mathbin{|\!\!\lfloor}, {}^c\_, \_^c, \partial_H, \rho_{\overline{\gamma}}$ a set of generators.

iii. Does a prime factorisation theorem for parallel composition hold in $\mathcal{G}_1(CA)$ (cf. [MM93]).

iv. For which recursive process specifications over $\mathcal{G}_{ser}(CA, \overline{\gamma}, C)$ can the communication operator be eliminated in favor of the causal state operator.

4. CONCLUSION.

In this paper, we have shown examples that expressions of the form $\partial_H \circ \rho_{\bar{\gamma}}(P_1 \parallel ... \parallel P_n)$ can be written in the form $\lambda^C_{T,V}(Q_1 \parallel ... \parallel Q_m)$. The former is a system specification involving actions that cannot occur ('halves' of communications, like $r_1(d)$), the latter is a service specification, only involving actions that actually happen. In this sense the former is more abstract. The latter is closer to reality, but is less modular. As a typical example, we considered a buffer of capacity 3.

Replacing communication by causality is a step forward in terms of understanding a system in terms of its externally visible behaviour only.

REFERENCES.

[BB88] J.C.M. BAETEN & J.A. BERGSTRA, *Global renaming operators in concrete process algebra,* Inf. & Comp. 78 (3), 1988, pp. 205-245.

[BB91] J.C.M. BAETEN & J.A. BERGSTRA, *Real time process algebra,* in Formal Aspects of Computing 3 (2), 1991, pp. 142-188.

[BB93] J.C.M. BAETEN & J.A. BERGSTRA, *Non interleaving process algebra,* in Proc. CONCUR'93, Hildesheim (E. Best, ed.), Springer LNCS 715, 1993, pp. 308-323.

[BW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in TCS 18, Cambridge University Press 1990.

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

[BK86] J.A. BERGSTRA & J.W. KLOP, *Verification of an alternating bit protocol by means of process algebra,* in: Math. Methods ov Spec. and Synthesis of Software Systems '85 (W. Bibel & K.P. Jantke, eds.), Springer LNCS 215, 1986, pp. 9-23.

[BKT85] J.A. BERGSTRA, J.W. KLOP & J.V. TUCKER, *Process algebra with asynchronous communication mechanisms,* in Proc. Seminar on Concurrency (S.D. Brookes, A.W. Roscoe & G. Winskel, eds.), Springer LNCS 197, 1985, pp. 76-95.

[BDH92] E. BEST, R. DEVILLERS & J.G. HALL, *The Petri box calculus: a new causal algebra with multilabel communication,* in: Advances in Petri Nets 92 (G. Rozenberg, ed.), Springer LNCS 609, 1992, pp. 21-69.

[CH89] I. CASTELLANI & M. HENNESSY, *Distributed bisimulations,* JACM 10, 1989, pp. 887-911.

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

[GOL88] U. GOLTZ, *On representing CCS programs by finite Petri nets,* in Proc. MFCS 88, Springer LNCS 324, 1988, pp. 339-350.

[JPZ91] W. JANSSEN, M. POEL & J. ZWIERS, *Action systems and action refinement in the development of parallel systems,* in: Proc. CONCUR'91 (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 298-316.

[KIE89] A. KIEHN, *Distributed bisimulations for finite CCS,* report 7/89, University of Sussex 1989.

[MIL93] R. MILNER, *Elements of interaction (Turing Award lecture),* Comm. of the ACM 36 (1), 1993, pp. 78-89.

[MM93] R. MILNER & F. MOLLER, *Unique decomposition of processes,* Theor. Comp. Sci. 107 (2), 1993, pp. 357-363.

[NT84] M. NIELSEN & P.S. THIAGARAJAN, *Degrees of non-determinism and concurrency: a Petri net view,* in: Proc. 5th FST&TCS (M. Joseph & R. Shyamasundar, eds.), Springer LNCS 181, 1984, pp. 89-118.

[PET80] C. PETRI, *Introduction to general net theory,* in: Net Theory and Applications (W. Brauer, ed.), Springer LNCS 84, 1980, pp. 1-19.

[POM86] L. POMELLO, *Some equivalence notions for concurrent systems. An overview,* in: Advances in Petri Nets 1985 (G. Rozenberg, ed.), Springer LNCS 222, 1986, pp. 381-400.