Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Bergstra, J.W. Klop

Fair FIFO queues satisfy an algebraic criterion
for protocol correctness

Department of Computer Science Report CS-R8405 March

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

FAIR FIFO QUEUES SATISFY AN ALGEBRAIC CRITERION FOR PROTOCOL CORRECTNESS

J.A. BERGSTRA, J.W. KLOP
Centre for Mathematics and Computer Science, Amsterdam

An algebraic criterion for protocol correctness is presented, as well as a proof method for establishing the criterion. As an example we consider FIFO queues with unbounded capacity.

bgF11, bgF12, bgF32, bgF43

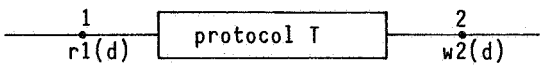
1980 MATHEMATICS SUBJECT CLASSIFICATION: 68B10, 68C01, 68D25, 68F20.
1982 CR. CATEGORIES: F.1.1, F.1.2, F.3.2, F.4.3.
KEY WORDS & PHRASES: process algebra, queue, communication protocol, verification.
NOTE: This report will be submitted for publication elsewhere.

Report CS-R8405
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

INTRODUCTION

The purpose of this paper is to formulate a criterion for communication protocol validity within the framework of process algebra, and to test it on some simple examples.

To set the stage of the subsequent discussion, let D be a finite alphabet of data, and consider this picture:



The task of T is to transmit a stream of data from site 1 to site 2. We write r1(d) for the action by which T reads datum d in 1, and w2(d) for the action by which T writes (outputs) datum d at 2.

Informally validity of T amounts to this:

(1) T will eventually write all and only data it has read,

(2) T will write the data in the same order as it has read them.

In general T will be a complex mechanism made up from several components that have to coöperate properly in order to ensure correct protocol behaviour.

There are several ways to make a mathematical model of T. One way is to view T as a process in the sense of Milner [5]. We will use the related framework of process algebra. T is now seen as a process over a set of atomic actions A(T). Here A(T) contains four types of actions as listed below:

(i) read actions r1(d) for $d \in D$

(ii) write actions w2(d) for $d \in D$

(iii) internal actions $j \in I_T$

(iv) deadlock: δ.

It is possible that the set of internal actions I_T is empty, and it is possible that T does not use δ (this is desirable indeed).

1.1. Input streams and output streams.

With $\vec{d}$ we denote a finite or infinite sequence of values from D, i.e. $\vec{d} \in D^* \cup D^\omega$.

An input stream for T is a process that generates successive inputs

for T. Let $\vec{d} = (d1,\ldots,dn) \in D^*$. Then the input stream $p_1(\vec{d})$ corresponding to d is the process:

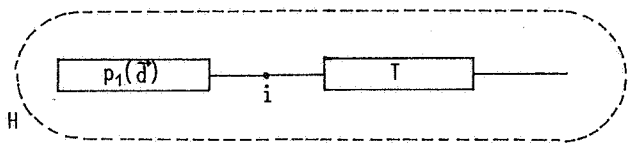
$$wl(d1).wl(d2)....wl(dn).$$

Here wl(d) is the write action at location 1. It is understood that the "reads" and "writes" at various locations communicate:

$$\boxed{wl(d) \mid rl(d) = i}$$

This means that an action wl(d) can only be performed simultaneously with an action rl(d), and together these actions are fused into an (internal) action i.

Providing T with an input stream amounts to the construction of a simple network:



Here: $A(p_1(\vec{d})) \subseteq \{wl(d) \mid d \in D\}$

$H = \{wl(d) \mid d \in D\} \cup \{rl(d) \mid d \in D\}$

$I = \{i\} \cup I_T$.

The full set of atomic actions in which this takes place is A:

$$A = A(p_1(\vec{d})) \cup \{i,\delta\} \cup A(T).$$

The network is described by

$$\partial_H(T \| p_1(\vec{d})).$$

This is: T and $p_1(\vec{d})$ operate in parallel under the constraint that the communication actions in H find a matching action within $T \| p_1(\vec{d})$.

For the later sections we need a precise definition of $p_1(\vec{d})$:

(i) if $\vec{d} = \varepsilon$ then $p_1(\vec{d}) = \tau$, where τ is the silent action.

(ii) if $\vec{d} = (d1,\ldots,dn)$ then $p_1(\vec{d}) = wl(d1)....wl(dn)$.

(iii) if $\vec{d} = (d1,d2,\ldots)$ then $p_1(\vec{d}) = wl(d1).w2(d2). \ldots$

We write $A_\tau = A \cup \{\tau\}$. In the universe of processes over A_τ computations involving abstraction can be carried out.

Quite similar to the $p_1(\vec{d})$ we define the <u>output streams</u> $p_2(\vec{d})$ with actions w2(d) instead of wl(d). Note that $A(p_2(\vec{d})) \subseteq A(T)$, whereas $A(p_1(\vec{d})) \cap A(T) = \emptyset$.

1.2. <u>An algebraic criterion for the correctness of T.</u>

In the terminology of 1.1, with the understanding that $.|.: A \times A \to A$ yields δ except for $wl(d) \mid rl(d) = i$, we obtain the following algebraic criterion for T's correctness:

C(T): For all $\vec{d} \in D^* \cup D^\omega$:

$$\tau_I \partial_H(T \| \tau.p_1(\vec{d}).\delta) = \tau.p_2(\vec{d}).\delta$$

<u>Explanation</u>:

(i) Apart from the τ's and δ's which will be commented below, the identity says that for any input stream $p_1(\vec{d})$, if it is composed with T, and all actions except those visible at the output site (2) are abstracted away, the corresponding output stream is obtained.

Stated in different words: seen from location 2, the system $\partial_H(T \| \tau.p_1(\vec{d}).\delta)$ is just an output stream. Note that $I = \{i\} \cup I_T$ contains all actions that are not visible at 2.

(ii) About the τ and δ pre- and postfixes: first note that in general

$$\tau_I \partial_H(x \| \tau y) = \tau.\tau_I \partial_H(x \| y)$$

$$\tau_I \partial_H(x \| y\delta) = \tau_I \partial_H(x \| y).\delta$$

and in particular

$$\tau_I \partial_H(T \| \tau.p_1(\vec{d}).\delta) = \tau.\tau_I \partial_H(T \| p_1(\vec{d})).\delta .$$

Now observe that it is unrealistic to expect

$$\tau_I \partial_H(T \| p_1(\vec{d})) = p_2(\vec{d})$$

because $\partial_H(T||p_1(\vec{d}))$ will in general have to perform some internal actions before yielding its first output. I.e.

$$\tau_I\partial_H(T||p_1(\vec{d})) = \tau.p_2(\vec{d})$$

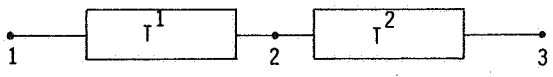
is the best we can hope for. For reasons of symmetry we write this as follows:

$$\tau_I\partial_H(T||\tau.p_1(\vec{d})) = \tau.p_2(\vec{d}).$$

(iii) A similar argument (based on Koomen's fair abstraction rule) explains that reasons of symmetry impose postfixing δ's on both sides.

1.4. The composition of correct protocols.

Let T^1, T^2 be two correct protocols that transmit data from D:



Let $\alpha(T^1) = r1(D) \cup w2(D) \cup \{\delta\} \cup I_{T1}$, and

$\alpha(T^2) = r2(D) \cup w3(D) \cup \{\delta\} \cup I_{T2}$.

We can decompose T^1 and T^2 sequentially as follows: let $H_2 = r2(D) \cup w2(D)$, and put

$$r2(d) \mid w2(d) = i$$

Furthermore, put

$$T^* = \partial_{H_2}(T^1||T^2).$$

Then T* is a protocol from 1 to 3, with alphabet

$$\alpha(T^*) = r1(D) \cup w3(D) \cup \{\delta,i\} \cup I_{T1} \cup I_{T2}.$$

Correctness of T* follows thus: let $I^* = \{i\} \cup I_{T1} \cup I_{T2} = I \cup I_{T2}$, and let $H^* = H \cup H_2$. Then:

$$\tau_{I^*}\partial_{H^*}(T^*||\tau.p_1(\vec{d}).\delta) =$$

$$\tau_{I^*}\partial_{H^*}(\partial_H(T^1||T^2)||\tau.p_1(\vec{d}).\delta) =$$

$$\tau_{I^*}\partial_{H^*}(T^1 \,||\, T^2 \,\|\, \tau.p_1(\vec{d}).\delta) =$$

$$\tau_{I^*}\partial_{H^*}((T^1 \,||\, \tau.p_1(\vec{d}).\delta) \,\|\, T^2) =$$

$$\tau_{I^*}\partial_{H^*}(\partial_H(T^1 \,||\, \tau.p_1(\vec{d}).\delta) \,\|\, T^2) =$$

$$\tau_{I^*}\partial_{H^*}(\tau_I\partial_H(T^1 \,||\, \tau.p_1(\vec{d}).\delta) \,\|\, T^2) =$$

$$\tau_{I^*}\partial_{H^*}(\tau.p_2(\vec{d}).\delta \,||\, T^2) =$$

$$\tau.p_3(\vec{d}).\delta \ .$$

Remark. This proof requires more axioms and rules about process algebra than just those that are presented in Section 2.

1.4. A strategy for proving C(T).

In general T will be a rather complex process with many internal states. Let $T(.)$: S $\longrightarrow$ states of T be a parametrisation of a subset of the states of T such that in particular $T(s_0) = T$ for some $s_0 \in S$. Because each $T(s)$ is some state of T, we may postulate the existence of a mapping

$$\text{cont: } S \longrightarrow D^*$$

where cont(s) computes the "contents" of s, i.e. the string of values from D that T in state T(s) will eventually generate as outputs when no new inputs are given at all. Clearly $\text{cont}(s_0) = \varepsilon$.

C(T) then follows from the following stronger condition C'(T):

> C'(T): For all $s \in S$ and $\vec{d} \in D^* \cup D^\omega$:
>
> $$\tau_I\partial_H(T(s) \,\|\, \tau.p_1(\vec{d}).\delta) = \tau.p_2(\text{cont}(s)*\vec{d}).\delta$$

This formulation of the criterion clarifies how to deal with internal states of T (in principle).

Next we have to take into account that some kind of induction on the number of outputs that are produced may be necessary. Let $E = \{\delta\} \cup \{ w2(d) \mid d \in D \}$. Then the projection operators π_E^n (for $n \in \omega$) cut off each branch of a process

after the n-th occurrence of an action in E. In order to prove C'(T) it suffices to show C"(T):

C"(T): For all $s \in S$, $\vec{d} \in D^* \cup D^\omega$, $n \in \omega$:

$$\pi_E^n(\tau_I \partial_H(T(s) \,||\, \tau.p_1(\vec{d}).\delta)) = \pi_E^n(\tau.p_2(cont(s)*\vec{d}).\delta)$$

To establish C"(T) some induction on n may be of use.

2. PROCESS ALGEBRA

In this section we will state axioms and rules of process algebra inasmuch as these are needed for the technical discussion of some examples in Sections 3 and 4.

2.1. $\underline{ACP}_\tau(\pi, \tau^E)$.

Let A be a set of atomic actions and $.|.: A \times A \to A$ a communication function which is commutative and associative and for which $\delta \in A$ acts as a zero. A_τ denotes $A \cup \{\tau\}$, where τ is Milner's silent action. The signature of operations that we will use is this:

$+$	*alternative composition (sum)*
$.$	*sequential composition (product)*
$\|\|$	*parallel composition (merge)*
$\mathbin{\|\!\_}$	*left-merge*
$\|$	*communication merge*
∂_H	*encapsulation*
τ_I	*abstraction*
δ	*deadlock*
τ	*silent action*
π_E^n	*n-th E-projection*
τ_I^E	*E-delayed abstraction*

Here E,H,I are sets of atomic actions such that

$$H \subseteq A \qquad H \cap I = \emptyset$$
$$I \subseteq A - \{\delta\} \qquad H \cap E = \emptyset$$
$$E \subseteq A \qquad E \cap I = \emptyset.$$

In Table 1 we have displayed important axioms that fix the interrelations of the operators on all finite processes (i.e. processes denoted by closed terms over the signature).

Various important subsignatures have been studied earlier together with restricted axiom tables, namely

PA: $+, \cdot, \|, \lfloor\!\lfloor$

ACP: $+, \cdot, \|, \lfloor\!\lfloor, \partial_H, \delta$

ACP_τ: $+, \cdot, \|, \lfloor\!\lfloor, |, \partial_H, \tau_I, \delta, \tau$

The axioms of PA are A1-5, $x\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x$, and CM2-4. (See Table 1 next page.)

The axioms of ACP are those in the left-upper quadrant of Table 1.

The axioms of ACP_τ are in the upper half of Table 1.

All axioms concerning τ are in the right half of Table 1.

In Table 1, a,b,c,e range over A. Further, in the P- and PT-axioms, $n \geqslant 1$.

Remark. ACP was first described in Bergstra & Klop [2], ACP_τ was studied in depth in Bergstra & Klop [3]. The axioms A1-3, I1-3 are taken from Milner [5].

$x+y=y+x$	A1	$x\tau=x$	T1
$x+(y+z)=(x+y)+z$	A2	$\tau x+x=\tau x$	T2
$x+x=x$	A3	$a(\tau x+y)=a(\tau x+y)+ax$	T3
$(x+y)z=xz+yz$	A4		
$(xy)z=x(yz)$	A5		
$x+\delta=x$	A6		
$\delta x=\delta$	A7		
$a\|b=b\|a$	C1		
$(a\|b)\|c=a\|(b\|c)$	C2		
$\delta\|a=\delta$	C3		
$x\|\|y=x\lfloor\!\lfloor y+y\lfloor\!\lfloor x+x\|y$	CM1		
$a\lfloor\!\lfloor x=ax$	CM2	$\tau\lfloor\!\lfloor x=\tau x$	TM1
$(ax)\lfloor\!\lfloor y=a(x\|\|y)$	CM3	$(\tau x)\lfloor\!\lfloor y=\tau(x\|\|y)$	TM2
$(x+y)\lfloor\!\lfloor z=x\lfloor\!\lfloor z+y\lfloor\!\lfloor z$	CM4	$\tau\|x=\delta$	TC1
$(ax)\|b=(a\|b)x$	CM5	$x\|\tau=\delta$	TC2
$a\|(bx)=(a\|b)x$	CM6	$(\tau x)\|y=x\|y$	TC3
$(ax)\|(by)=(a\|b)(x\|\|y)$	CM7	$x\|(\tau y)=x\|y$	TC4
$(x+y)\|z=x\|z+y\|z$	CM8		
$x\|(y+z)=x\|y+x\|z$	CM9		
		$\partial_H(\tau)=\tau$	DT
		$\tau_I(\tau)=\tau$	TI1
$\partial_H(a)=a$ if $a\notin H$	D1	$\tau_I(a)=a$ if $a\notin I$	TI2
$\partial_H(a)=\delta$ if $a\in H$	D2	$\tau_I(a)=\tau$ if $a\in I$	TI3
$\partial_H(x+y)=\partial_H(x)+\partial_H(y)$	D3	$\tau_I(x+y)=\tau_I(x)+\tau_I(y)$	TI4
$\partial_H(xy)=\partial_H(x).\partial_H(y)$	D4	$\tau_I(xy)=\tau_I(x).\tau_I(y)$	TI5
$\pi_E^n(a)=a$	P1	$\pi_E^0(x)=\tau$	PI1
$\pi_E^1(e.x)=e$ if $e\in E$	P2	$\pi_E^n(\tau)=\tau$	PI2
$\pi_E^{n+1}(e.x)=e.\pi_E^n(x)$ if $e\in E$	P3	$\pi_E^n(\tau.x)=\tau.\pi_E^n(x)$	PI3
$\pi_E^n(a.x)=a.\pi_E^n(x)$ if $a\in A-E$	P4		
$\pi_E^n(x+y)=\pi_E^n(x)+\pi_E^n(y)$	P5		
		$\tau_I^E(a)=a$	TE1
		$\tau_I^E(\tau)=\tau$	TE2
		$\tau_I^E(e.x)=e.\tau_I(x)$ if $e\in E$	TE3
		$\tau_I^E(\tau.x)=\tau.\tau_I^E(x)$	TE4
		$\tau_I^E(b.x)=b.\tau_I^E(x)$ if $b\in A-E$	TE5
		$\tau_I^E(x+y)=\tau_I^E(x)+\tau_I^E(y)$	TE6

Table 1.

$ACP_\tau(\pi,\tau^E)$ yields all identities we want on finite processes. Many identities $t_1(x_1,\ldots,x_n) = t_2(x_1,\ldots,x_k)$ that hold for all finite processes are entirely plausible for infinite processes as well. Because such identities are in general unprovable from the given equational specification, we must add such identities in many cases. In Sections 3 and 4 we will need:

$$\tau_I\,\tau_I^E(x) = \tau_I(x)$$

$$\tau_I\,\pi_E^m(x) = \pi_E^m\,\tau_I(x)$$

In fact we know of no example of an identity $t_1(x_1,\ldots,x_k) = t_2(x_1,\ldots,x_k)$ which is provable for each instantiation with closed terms from $ACP_\tau(\pi,\tau^E)$ and which is implausible for processes in general. Some type of ω-rule might well be valid here.

Besides laws of an equational nature we will need some laws of a second-order character as well. In particular the recursive specification principle (RSP) and Koomen's fair abstraction rule (KFAR) are needed. These principles will now briefly be explained.

2.2. Recursive specification principle (RSP).

Let X,Y,X_i,Y_i $(i \in \omega)$ be variables for processes. We write $\underline{X}$ for $\{X_i \mid i \in \omega\}$ and $\underline{Y}$ for $\{Y_i \mid i \in \omega\}$. If Z is a collection of variables then t(Z) denotes an $ACP_\tau(\pi,\tau^E)$-term over Z.

Let $F \subseteq A$; we call the term t(Z) F-guarded if each variable in t(Z) is preceded by an atom in F which is not in the scope of some τ_I or τ_I^E operator. We call an equation X = t(Z) F-guarded if t(Z) is F-guarded.

DEFINITION. A recursive specification $S_F(X,\underline{X})$ is a collection of F-guarded equations (over $ACP_\tau(\pi,\tau^E)$)

$$X_i = t_i(\underline{X})$$

together with an equation

$$X = t(\underline{X}).$$

Remark. If p, p_i $(i \in \omega)$ satisfy $S_F(p; p_i \mid i \in \omega)$ then we want to view $S_F(X, X)$ as a specification of p involving auxiliary processes p_i $(i \in \omega)$.

Of course this definition includes the case of a finite specification.

The recursive specification principle RSP states that a recursive definition singles out a unique process (if any). In more formal notation:

$$\text{(RSP)}\quad \frac{S_F(X,X) \qquad S_F(Y,Y)}{X = Y}$$

2.2. Koomen's fair abstraction rule.

We will describe a simplified version of Koomen's rule which is of use in Section 4. In Bergstra & Klop [4] the rule is explained in full generality.

$$\text{(KFAR)}\quad \frac{X = i.X + Y}{\tau_I(X) = \tau.\tau_I(Y)} \qquad i \in I$$

This rule expresses the fact that the process X will not perform an infinite sequence of internal steps (due to fairness of the mechanism behind the choices that X makes).

For a different approach to fairness in processes we refer to De Bakker & Zucker [1].

3. EXAMPLE 1, A SINGLE BIT BUFFER

A single bit buffer is defined by the following recursion equation:

$$T = \sum_{d \in D} r1(d).w2(d).T$$

This is the simplest of all protocols. In [4] it has been shown within the

framework of process algebra that ABP, an example of the alternating bit protocol, is equivalent to T.

We will verify that T satisfies C"(T).

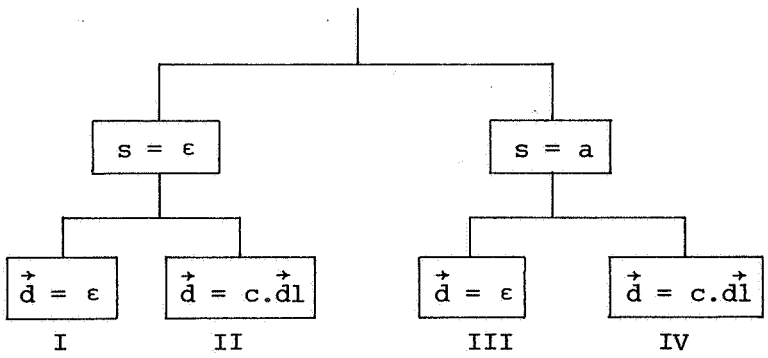
Let $S = \{\varepsilon\} \cup D$, $s_0 = \varepsilon$. We define the operators T(x) and cont(x) on S by

$$\begin{cases} T(\varepsilon) = T \\ T(d) = w2(d).T \\ cont(\varepsilon) = \varepsilon \\ cont(d) = d. \end{cases}$$

We will show that for all $s \in S$, for all $\vec{d} \in D^* \cup D^\omega$ and for all $n \in \omega$:

$$\pi_E^n(\tau_I \partial_H(T(s) \| \tau.p_1(\vec{d}).\delta)) = \pi_E^n(\tau.p_2(\vec{d}).\delta).$$

Using induction on n the case n = 0 is obvious and the case n = m + 1 leads to a case distinction:



All four cases are essentially straightforward. We consider case IV only.

$$\pi_E^{m+1} \tau_I \partial_H(T(a) \| \tau.p_1(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1} \tau_I \partial_H(T(a) \| p_1(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1} \tau_I w2(a).\partial_H(T \| p_1(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1} w2(a).\tau_I \partial_H(T \| p_1(\vec{d}).\delta) =$$

$$\tau.w2(a).\pi_E^m \tau_I \partial_H(T \| \tau.p_1(\vec{d}).\delta) =$$

$$\tau.w2(a).\pi_E^m(\tau.p_2(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1}(w2(a).\tau.p_2(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1}(p_2(a*\vec{d}).\delta) =$$

$$\pi_E^{m+1}(\tau.p_2(cont(a)*\vec{d}).\delta) \ .$$

4. FIFO QUEUES

Let $S = D^*$; for each $\vec{d} \in S$, $Q(\vec{d})$ denotes the queue that contains $\vec{d} = (d1,..,dn)$, in the sense that dn was its last input and d1 will be its next output.

The states $Q(\vec{d})$ of $Q = Q(\varepsilon)$ satisfy these recursion equations:

$$Q(\varepsilon) = \sum_{d\in D} rl(d).Q(d)$$

$$Q(a*\vec{d}) = \sum_{c\in D} rl(c).Q(a*\vec{d}*c) + w2(a).Q(\vec{d})$$

We will verify C"(T): for all $s \in D^*$, $n \in \omega$, $\vec{d} \in D^* \cup D^\omega$:

$$\pi_E^n \tau_I \partial_H(Q(s) \| \tau.p_1(\vec{d}).\delta) = \pi_E^n(\tau.p_2(\vec{d}).\delta).$$

The proof involves a case distinction and two inductions which in turn lead to case distinctions. The various subcases are explained in a diagram (see next page).

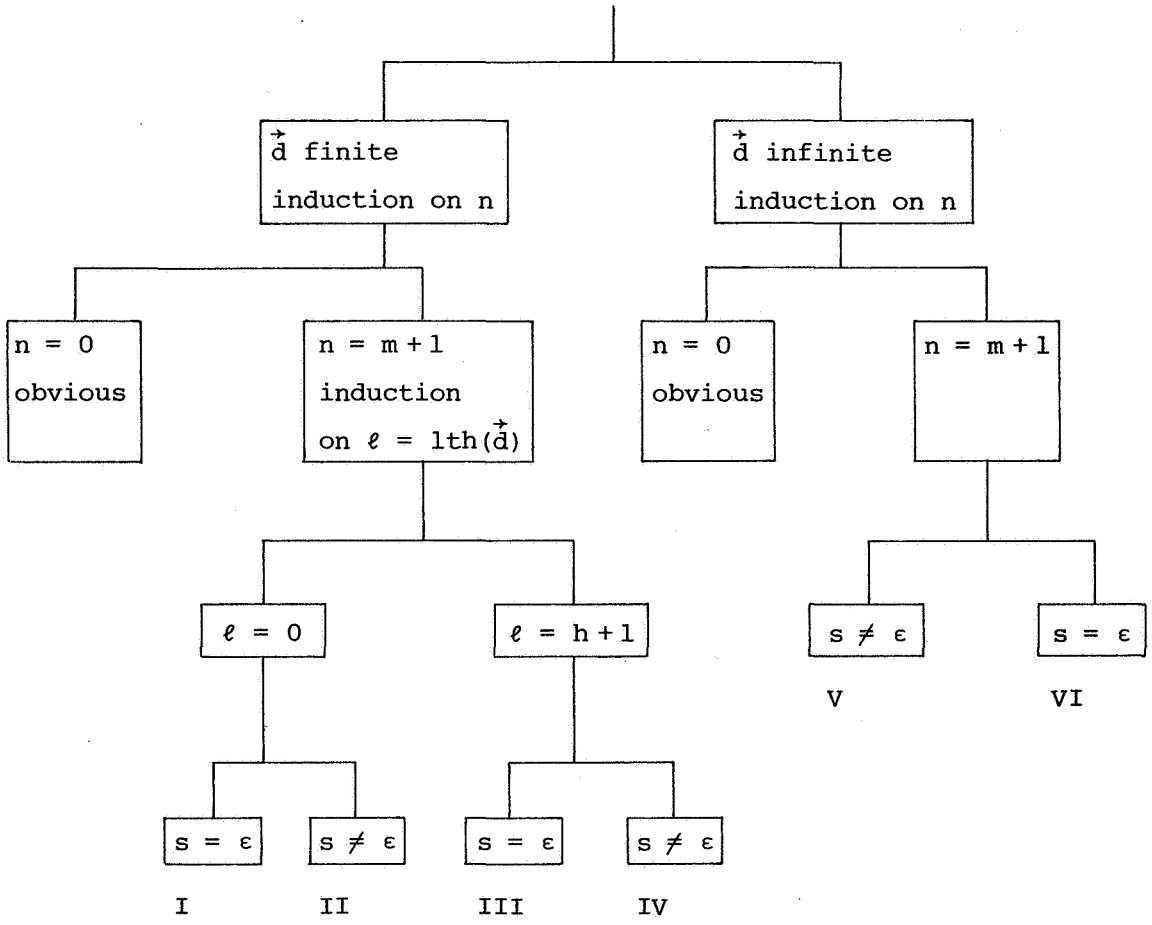
We will use the following notational conventions below:

(i) $s = b1*...*bk = b1*s'$ unless $s = \varepsilon$;

(ii) if $lth(\vec{d}) < \omega$ then $\vec{d} = (d1,...,dn) = d1.\vec{d}'$

(iii) if $lth(\vec{d}) = \omega$ then $\vec{d} = (d1,d2,...) = d1.\vec{d}'$.

(Diagram of case distinctions:)



Case I.

$$\pi_E^{m+1}\tau_I\partial_H(Q(\varepsilon)\,||\,\tau.\tau.\delta) =$$

$$\tau.\pi_E^{m+1}(Q(\varepsilon)\,||\,\delta) = \tau.\delta = \pi_E^{m+1}(\tau.p_2(\varepsilon).\delta).$$

Case II.

$$\pi_E^{m+1}\tau_I\partial_H(Q(s)\,||\,\tau.\tau.\delta) =$$

$$\tau.\pi_E^{m+1}\tau_I\partial_H(Q(bl*s')\,||\,\delta) =$$

$$\tau.w2(bl).\pi_E^{m}\tau_I\partial_H(Q(s')\,||\,\tau.\delta) =$$

$$\tau.w2(b2).\pi_E^m(\tau.p_2(s').\delta) =$$

$$\tau.\pi_E^{m+1}(w2(b2).p_2(s').\delta) =$$

$$\pi_E^{m+1}(\tau.p_2(s).\delta) \ .$$

Case III.

$$\pi_E^{m+1}\tau_I\partial_H(Q(\varepsilon)||\tau.p_1(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1}\tau_I\partial_H(Q(\varepsilon)||w1(d1).p_1(\vec{d'}).\delta) =$$

$$\tau.\pi_E^{m+1}\tau_I(i.\partial_H(Q(d1)||p_1(\vec{d'}).\delta)) =$$

$$\tau.\pi_E^{m+1}\tau_I\partial_H(Q(d1)||\tau.p_1(\vec{d'}).\delta) =$$

$$\tau.\pi_E^{m+1}(\tau.p_2(d1*\vec{d'}).\delta) =$$

$$\pi_E^{m+1}(\tau.p_2(\vec{d}).\delta) \ .$$

Case IV.

$$\pi_E^{m+1}\tau_I\partial_H(Q(s)||\tau.p_1(\vec{d}).\delta) =$$

$$\tau.\pi_E^{m+1}\tau_I\{w2(b1).\partial_H(Q(s')||\tau.p_1(\vec{d}).\delta) +$$

$$+ i.\partial_H(Q(s'*d1)||\tau.p_1(\vec{d'}).\delta)\} =$$

$$\tau.\{w2(b1).\pi_E^m\tau_I\partial_H(Q(s')||\tau.p_1(\vec{d}).\delta) +$$

$$+ \tau.\pi_E^{m+1}\tau_I\partial_H(Q(s*d1)||\tau.p_1(\vec{d'}).\delta)\} =$$

$$\tau.\{w2(b1).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) +$$

$$+ \tau.\pi_E^{m+1}(\tau.p_2(s*d1*\vec{d'}).\delta)\} =$$

$$\tau.\{w2(b1).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) +$$

$$+ \tau.w2(b1).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta)\} =$$

$$\tau.w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) =$$

$$\pi_E^{m+1}(\tau.p_2(s'*\vec{d}).\delta) .$$

Case V. We need some special notation. We write for $j\varepsilon\omega$:

$$\vec{d} = dl*\ldots*dj*d(j+1)*d(j+2)*\cdots$$

and

$$Z_j = Q(s*dl*\ldots*dj)\,||\,(\tau.p_1(d(j+1)*d(j+2)*\ldots).\delta).$$

Then

$$\pi_E^{m+1}\tau_I\partial_H(Q(\varepsilon)\,||\,\tau.p_1(\vec{d}).\delta) = \pi_E^{m+1}\tau_I\partial_H(Z_0).$$

It obviously suffices to show that:

$$\pi_E^{m+1}\tau_I\partial_H(Z_0) = \pi_E^{m+1}(\tau.p_2(s*\vec{d}).\delta)$$

Note that for all j:

$$\partial_H(Z_j) = w2(bl).\partial_H(s'*dl*\ldots*dj\,||\,\tau.p_1(d(j+1)*\ldots).\delta) + i.\partial_H(Z_{j+1}).$$

Applying τ_I^E on both sides we find:

$$\tau_I^E\partial_H(Z_j) = w2(bl).\tau_I\partial_H(s'*dl*\ldots*dj\,||\,\tau.p_1(d(j+1)*\ldots).\delta) +$$

$$+ i.\tau_I^E\partial_H(Z_{j+1}).$$

Applying π_E^{m+1} on both sides we obtain

$$\pi_E^{m+1}\tau_I^E\partial_H(Z_j) =$$

$$w2(bl).\pi_E^m\tau_I\partial_H(s'*dl*\ldots*dj\,||\,\tau.p_1(d(j+1)*\ldots).\delta) + i.\pi_E^{m+1}\tau_I\partial_H(Z_{j+1}) =$$

$$w2(bl).\pi_E^m(\tau.p_2(s'*dl*\ldots*dj\,||\,\tau.p_1(d(j+1)*\ldots).\delta) + i.\pi_E^{m+1}\tau_I\partial_H(Z_{j+1}) =$$

$$w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) + i.\pi_E^{m+1}\tau_I\partial_H(Z_{j+1}).$$

If we write

$$U_j = \pi_E^{m+1} \tau_I^E \partial_H(Z_j)$$

then the U_j solve the system of equations

$$X_j = w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) + i.X_{j+1}.$$

Let X be the solution of

$$X = w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) + i.X$$

then, by RSP for all j, $U_j = X$. In particular $U_0 = X$ and therefore:

$$\pi_E^{m+1} \tau_I^E \partial_H(Z_0) = w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta) + i.\pi_E^{m+1} \tau_I^E \partial_H(Z_0).$$

At this point one may apply Koomen's fair abstraction rule:

$$\tau_I \pi_E^{m+1} \tau_I^E \partial_H(Z_0) = \tau.\tau_I(w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta))$$

Simplifying both sides:

$$\pi_E^{m+1} \tau_I \tau_I^E \partial_H(Z_0) = \tau.w2(bl).\pi_E^m(\tau.p_2(s'*\vec{d}).\delta)$$

$$\pi_E^{m+1} \tau_I \partial_H(Z_0) = \pi_E^{m+1}(\tau.p_2(s*\vec{d}).\delta)$$

Conclusions.

We have formulated an algebraic criterion C(T) for communication protocol correctness. In Sections 3 and 4, C(T) has been verified for two examples.

Obviously these verifications are quite involved, and seem to constitute an unbearable overhead given the simplicity of these examples.

The motivation for this type of work is this: protocol verification is a quite sophisticated matter and application of process algebra in this area might be useful. However it is likely that a significant amount of special purpose mathematics is required to obtain such applications. Theoretical work on simple examples can generate such mathematics. The example of Section 4 for instance has lead to the definition of the operator τ_I^E.

Clearly there is much work ahead in trying to specify and verify increasingly complex protocols. In view of [6] the correctness criterion itself is a topic for research just as well.

REFERENCES

[1] DE BAKKER, J.W. & J.I. ZUCKER, *Compactness in semantics for merge and fair merge*, Report IW 238/83, Mathematisch Centrum, Amsterdam 1983.

[2] BERGSTRA, J.A. & J.W. KLOP, *Process algebra for communication and mutual exclusion*, Report IW 218/83, Mathematisch Centrum, Amsterdam 1983.

[3] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes with abstraction*, Report CS-R8403, Centrum voor Wiskunde en Informatica, Amsterdam 1984.

[4] BERGSTRA, J.A. & J.W. KLOP, *Verification of an alternating bit protocol by means of process algebra*, Report CS-R8404, Centrum voor Wiskunde en Informatica. Amsterdam 1984.

[5] MILNER, R., *A Calculus of Communicating Systems*, Springer LNCS 92, 1980.

[6] YEMENI, Y. & J.F. KUROSA, *Can current protocol verification techniques quarantee correctness?* Computer Networks, Vol.6, No.6 (1982),p.377-381.