VERIFICATION OF AN ALTERNATING BIT PROTOCOL BY MEANS OF PROCESS ALGEBRA

J.A. BERGSTRA, J.W. KLOP

Centre for Mathematics and Computer Science

P.O.Box 4079, 1009 AB Amsterdam, The Netherlands

We verify a simple version of the alternating bit protocol in the system ACP_τ (Algebra of Communicating Processes with silent actions) augmented with Koomen's fair abstraction rule.

INTRODUCTION

Let D be a finite set of data. These data are to be transmitted through an unreliable medium from location 1 to location 2, by means of a transmission protocol T.

With r1(d) we denote the act of reading datum d at location 1, whereas w2(d) denotes the act of writing value d at location 2. The external (higher level) specification of the behaviour of T is this:

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

From its initial state T is enabled to read any $d \in D$, thereafter T will write d at 2 and subsequently return to its initial state.

A very interesting mechanism to implement T is the alternating bit protocol (from [2]). This protocol turns out to be sufficiently complicated to serve as a test case for protocol verification methods (see HAILPERN & OWICKI [7] and LAMPORT [8] for instance).

We will present a description and verification of ABP (the alternating bit protocol), in terms of process algebra. Our presentation makes extensive use of ACP_τ, Algebra of Communicating Processes with silent actions, as well as of ideas by C.J. Koomen from Philips Research.

The advantage of process algebra in contrast to techniques based on temporal logic and Hoare-style verification is mainly that the entire verification is done in terms of calculations on the protocol itself. Both safety and liveness are simultaneously dealt with in the equational calculus of process algebra.

The structure of this note is as follows:

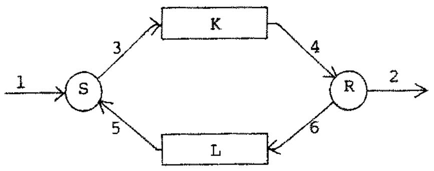
1. Explanation of the architecture of ABP.
2. Axioms and rules of process algebra.
3. Verification of ABP.

This work was sponsored in part by ESPRIT contract 432 METEOR.

Remark. It must be said that ABP as explained here is only one of the many variations on the same theme, and among these a rather simple one. Process algebra is well suited to specify individual protocols; at present the specification of classes of protocols is not supported by process algebra. For other issues of a philosophical nature we refer to [10] and [11].

1. ARCHITECTURE OF ABP

1.1. The protocol can be visualised as follows:



There are four components:

S: sender. S reads data d at 1 ($d \in D$), and communicates the data to channel K until an acknowledgement has been received via channel L.

K: data transmission channel. K communicates data in $D0 \cup D1$ ($Di = \{di \mid d \in D\}$), and may communicate these correctly or communicate an error value e. K is supposed to be fair in the sense that it will not produce an infinite consecutive sequence of error outputs.

R: receiver. R receives data from K, outputs them at 2 and sends back acknowledgements via L.

L: acknowledgement transmission channel. The task of L is to communicate boolean values from R to S. The channel L may yield error outputs but is also supposed to be fair.

The components S,K,R and L are processes. The protocol T is described by

$$\partial_H(S\|K\|R\|L).$$

Here $\|$ denotes parallel composition and ∂_H encapsulates $S\|K\|R\|L$ by requiring that no external processes may interfere in the communications at ports 3,4,5 and 6.

In order to obtain an abstract view of the protocol the operator τ_I is applied, which replaces internal actions (in I) by the silent action τ. Thus:

$$\boxed{T = \tau_I \partial_H(S\|K\|R\|L)}$$

Verification amounts to a proof that this T satisfies the equation

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

1.2. Structure of the components of ABP.

1.2.1. Data and actions.

D is the finite set of data that is to be transmitted by ABP. For $d \in D$, d0 and d1 are new data, obtained by appending 0 resp. 1 to d. We write:

$$D0 = \{d0 \mid d \in D\}$$
$$D1 = \{d1 \mid d \in D\}$$
$$\mathbb{D} = D \cup D0 \cup D1 \cup \{0,1,e\}.$$

$\mathbb{D}$ is the set of data that occur as parameter of atomic actions.

For $t \in \{1,...,6\}$ there are read and write actions:

rt(a), read $a \in \mathbb{D}$ at t

wt(a), write $a \in \mathbb{D}$ at t.

Here $t \in \{1,...,6\}$ is called a port (or location, but we prefer port). Communication takes place at ports only:

$$rt(a) \mid wt(a) = j,$$

where j is an internal action. Another kind of internal action is i. It corresponds to internal choices made by K and L. The entire alphabet A of proper actions is then as follows:

$$A = \{rt(a) \mid 1 \leqslant t \leqslant 6,\ a \in \mathbb{D}\} \cup \{wt(a) \mid 1 \leqslant t \leqslant 6,\ a \in \mathbb{D}\} \cup \{i,j,\delta\}.$$

The communication function $.|. : A \times A \to A$ yields δ (deadlock or failure) except in the case mentioned before: $rt(a) \mid wt(a) = j$.
Of course the abstraction operator will introduce Milner's silent action τ and the universe of discourse consists of the processes over $A_\tau = A \cup \{\tau\}$.
Furthermore H, the set of subatomic (or communication) actions is:

$$\bigcup_{t \in \{3,4,5,6\}} \bigcup_{a \in \mathbb{D}} \{rt(a), wt(a)\},$$

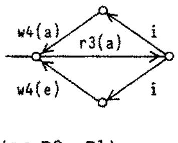
and I, the set of internal actions is just $\{i,j\}$.

1.2.2. The individual components.

We will first give the well-known state transition diagrams (or 'process graphs') for S,K,L and R. Here a node is a state and an arrow denotes an action (i.e. state transition of the process). Both state and actions can be parametrised by data.

Channels:

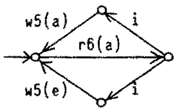
K:



(a∈ D0 ∪ D1)

$$\boxed{K = \sum_{a \in D0 \cup D1} r3(a).(i.w3(a) + i.w3(e)).K}$$

L:

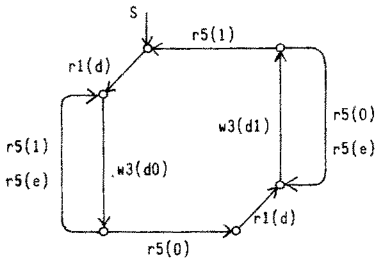


$$\boxed{L = \sum_{a \in \{0,1\}} r6(a).(i.w5(a) + i.w5(e)).L}$$

Note that K and L, after receiving input, have a nondeterministic choice, by doing one of both i actions.

At the level of this equational specification of K and L fairness is not yet mentioned. Fairness will come in when abstraction is applied to remove the i's.

Sender:



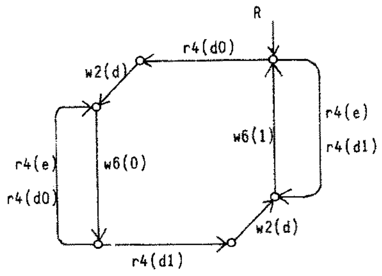
$$S = S^0.S^1.S$$

$$S^n = \sum_{d \in D} r1(d).S^n_d \qquad (n = 0,1)$$

$$S^n_d = w3(dn).U^n_d$$

$$U^n_d = (r5(1-n) + r5(e)).S^n_d + r5(n)$$

Receiver:



$$R = R^1.R^0.R$$

$$R^n = \left(\sum_{d\in D} r4(dn) + r4(e)\right).w6(n).R^n +$$

$$+ \sum_{d\in D} r4(d(1-n)).w2(d).w6(1-n)$$

$(n = 0,1)$

2. PROCESS ALGEBRA

2.1. ACP_τ.

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 δ acts as a zero.
A_τ denotes $A\cup\{\tau\}$; τ is the silent action, that results from application of the abstraction operator.

The signature of operations of processes that we will use is this:

$+$	*alternative composition (sum)*
$\cdot$	*sequential composition (product)*
$\|\|$	*parallel composition (merge)*
$\lfloor\!\lfloor$	*left-merge*
$\|$	*communication merge*
∂_H	*encapsulation*
τ_I	*abstraction*
δ	*deadlock/failure*
τ	*silent action*

Table 1.

An ACP_τ algebra is an algebra of the above signature (where $|$ extends the communication function on atoms) and which satisfies the axioms in Table 2. Here $H \subseteq A$, $I \subseteq A$, $\delta \notin I$ and a,b,c range over A.

ACP_τ

$x + y = y + x$	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		
$a \mid b = b \mid a$	C1		
$(a \mid b) \mid c = a \mid (b \mid c)$	C2		
$\delta \mid a = \delta$	C3		
$x \Vert y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	CM1		
$a \mathbin{\lfloor\!\lfloor} x = ax$	CM2	$\tau \mathbin{\lfloor\!\lfloor} x = \tau x$	TM1
$(ax) \mathbin{\lfloor\!\lfloor} y = a(x \Vert y)$	CM3	$(\tau x) \mathbin{\lfloor\!\lfloor} y = \tau(x \Vert y)$	TM2
$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	CM4	$\tau \mid x = \delta$	TC1
$(ax) \mid b = (a \mid b)x$	CM5	$x \mid \tau = \delta$	TC2
$a \mid (bx) = (a \mid b)x$	CM6	$(\tau x) \mid y = x \mid y$	TC3
$(ax) \mid (by) = (a \mid b)(x \Vert y)$	CM7	$x \mid (\tau y) = x \mid y$	TC4
$(x + y) \mid z = x \mid z + y \mid z$	CM8		
$x \mid (y + z) = x \mid y + x \mid z$	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

Table 2.

ACP_τ algebras satisfy the combinatorial identities shared by finite processes. In order to deal with infinite processes we will further assume that the following second order principles and rules are satisfied in the process algebra in which we model ABP, the alternating bit protocol.

I.	Recursive specification principle (RSP)
II.	Koomen's fair abstraction rule (KFAR)
III.	Handshaking axiom (HA)
IV.	Expansion Theorem (ET)

We will explain I-IV below. First, however, we allow ourselves some methodological remarks.

Remark 1. At present it is not possible to provide a remotely complete axiomatisation of processes that is of use "in general". But the equational (sub)-systems ACP and ACP_τ are a fixed kernel. Here ACP consists of the axioms A1-7,C1-3,CM1-9,D1-4, i.e. the left column of Table 2.

Remark 2. The system ACP was introduced in [3], and ACP_τ was introduced in [4]. We view ACP_τ as a reformulation of the basic issues of Milner's CCS [9] Comments on the relation between ACP_τ and CCS are in [4].

Remark 3. Koomen's fair abstraction rule has been derived from an idea that C.J. Koomen and R. Schutten used in experimental work on protocol verification . At Philips Research Eindhoven they have developed a formula manipulation package based on CCS.

2.2. Explanation of the principles I,II,III,IV.

2.2.1. The recursive specification principle.

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

Let $E\subseteq A$. We call the term t(Z) E-guarded if each variable in t(Z) is preceded by an atom in E and t(Z) does not contain an operator τ_I. Here 'preceded' is defined thus: if t is a subterm (occurrence) in x and s in y, then t precedes s in x.y; likewise in $x \mathbin{\|\!\lfloor} y$ (but not in $x\|y$).

Example: $a.(X\|Y) + b.X$ and $a.X.(X\|Y) + \tau.(b \mathbin{\|\!\lfloor} X)$ are $\{a,b\}$-guarded, but $\tau.X + b.Y$, $\tau.X|aY$ and $(aX\|Y) \mathbin{\|\!\lfloor} bZ$ are not E-guarded for any E.

We call an equation X = t(Z) E-guarded if t(Z) is E-guarded.

Remark. The reason for excluding operators τ_I in an E-guarded term is that it is problematic to find a suitable definition of guardedness in the presence of such operators. Here 'suitable' refers to our wish to obtain unique processes (by RSP; see below) as solutions of systems of guarded equations. E.g. the following system has infinitely many solutions:

$$X = a.\tau_{\{b\}}(Y)$$
$$Y = b.\tau_{\{a\}}(X).$$

Namely, every $X = a.p$ with a,b not in the alphabet of p is a solution.

DEFINITION. A recursive specification $S_E(X;\mathcal{X})$ is a collection of E-guarded equations (over ACP):

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

together with an equation

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

Remark. If P,P_i $(i\in\omega)$ satisfy the system of equations $S_E(P;P_i|i\in\omega)$ then we want to view $S_E(X;X_i|i\in\omega)$ 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:

$$\boxed{(\mathrm{RSP})\quad \frac{S_E(X;\mathcal{X})\qquad S_E(Y,\mathcal{Y})}{X=Y}}$$

2.2.2. Koomen's fair abstraction rule (KFAR).

This rule allows to compute $\tau_I(X)$ for certain X, thereby expressing the fact that certain steps in I will be fairly scheduled in such a way that eventually a step outside I is performed. This is the formal description of KFAR:

$$\boxed{(\mathrm{KFAR})\quad \frac{\forall n\in\mathbb{Z}_k\quad X_n = i_n.X_{n+1} + Y_n \qquad (i_n \in I)}{\tau_I(X_n) = \tau.\tau_I(Y_0+\ldots+Y_{k-1})}}$$

Here $\mathbb{Z}_k = \{0,\dots,k-1\}$ and addition in subscripts works modulo k.

We illustrate the effect of KFAR in two simple examples:

(i) Suppose $X = i.X + a$ where $a \notin I$. Then an application of KFAR yields: $\tau_I(X) = \tau.a$. This expresses the fact that, due to some fairness mechanism, i resists being performed infinitely many times consecutively.

(ii) Let $Y = i.Y$, then $\tau_I(Y) = \tau.\delta$. To see this note that $Y = i.Y + \delta$ and apply KFAR.

For a different approach to fairness in processes we refer to DE BAKKER & ZUCKER [1].

2.2.3. Axioms of standard concurrency.

We will adopt the following axioms of 'standard concurrency'; all axioms (1)-(6) hold for finite processes from ACP$_\tau$. In [5] these axioms are proved simultaneously with induction on term formation; we will only need here axioms (3)-(6).

(1)	$(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} (y \| z)$
(2)	$(x \mid ay) \mathbin{\lfloor\!\lfloor} z = x \mid (ay \mathbin{\lfloor\!\lfloor} z)$
(3)	$x \mid y = y \mid x$
(4)	$x \| y = y \| x$
(5)	$x \mid (y \mid z) = (x \mid y) \mid z$
(6)	$x \| (y \| z) = (x \| y) \| z$

2.2.4. Handshaking axiom (HA).

The handshaking axiom expresses the fact that all communications are binary, i.e. work by means of handshaking.

$$\boxed{\text{(HA)} \quad x \mid y \mid z = \delta}$$

2.2.5. Expansion Theorem (ET).

This theorem, in the context of CCS due to MILNER [9] and for ACP$_\tau$ formulated in [4], can be shown for finite processes from ACP$_\tau$. (See [5].) The Expansion Theorem presupposes HA and the axioms of standard concurrency (except (1),(2)). The following notation is used: Let $X_1,\dots,X_k$ be processes. With X^i we denote merge of all X_n such that $n \in \{1,\dots,k\} - \{i\}$. With $X^{i,j}$ we denote the merge of all X_n such that $n \in \{1,\dots,k\} - \{i,j\}$.

ET is then formulated as follows (for $k \geq 3$):

$$\boxed{(\mathrm{ET}) \qquad x_1 \| \ldots \| x_k = \sum_{1 \leqslant i \leqslant k} x_i \mathbin{⌊⌊} X^i + \sum_{1 \leqslant i < j \leqslant k} (x_i | x_j) \mathbin{⌊⌊} X^{i,j}}$$

ET is an indispensable tool for the calculation of terms of the form $x_1 \| \ldots \| x_k$. Essentially it is a generalisation of the axiom CM1 of ACP_τ.

3. A VERIFICATION OF ABP

Let $T^* = \sum_{d \in D} r1(d).w2(d).T^*$ and $T = \tau_I \partial_H(S \| K \| L \| R)$ in the notation of Section 1. Section 1 fixes a set of atomic actions A and a communication function on it.

Using ACP_τ + RSP + KFAR + HA + ET we will show: $T = T^*$. Stated differently:

$$\boxed{\tau_I \partial_H(S \| K \| L \| R) = \sum_{d \in D} r1(d).w2(d).\tau_I \partial_H(S \| K \| L \| R)}$$

For the proof we use the following notation:

$$(x_1 \| x_2 \| x_3 \| x_4) = \left(\begin{array}{c|c} x_1 & x_2 \\ \hline x_3 & x_4 \end{array}\right).$$

Using this notation we have:

$$T = \tau_I \partial_H \left(\begin{array}{c|c} S & K \\ \hline L & R \end{array}\right) = \tau_I \partial_H \left(\begin{array}{c|c} S^0.S^1.S & K \\ \hline L & R^1.R^0.R \end{array}\right).$$

For $b \in \{0,1\}$ we write

$$T^b(X,Y) = \tau_I \partial_H \left(\begin{array}{c|c} S^b.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y \end{array}\right).$$

In particular, $T = T^0(S,R)$.

<u>CLAIM:</u> $$T^b(X,Y) = \sum_{d \in D} r1(d).w2(d).\tau_I \partial_H \left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^b.Y \end{array}\right).$$

The claim proves $T = T^*$ as follows:

$$T = T^0(S,R) = \sum_{d \in D} r1(d).w2(d).\tau_I \partial_H \left(\begin{array}{c|c} S^1.S & K \\ \hline L & R^0.R \end{array}\right) =$$

$$\sum_{d\in D} r1(d).w2(d).\tau_I\partial_H\left(\begin{array}{c|c} S^1.S^0.S^1.S & K \\ \hline L & R^0.R^1.R^0.R\end{array}\right) =$$

$$\sum_{d\in D} r1(d).w2(d).T^1(S^1.S,\ R^0.R) =$$

$$\sum_{d\in D} r1(d).w2(d).\ \sum_{a\in D} r1(a).w2(a).\tau_I\partial_H\left(\begin{array}{c|c} S^0.S^1.S & K \\ \hline L & R^1.R^0.R\end{array}\right) =$$

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

Thus T satisfies an (A-guarded) recursion equation which is also satisfied by T*. It follows by RSP that $T = T^*$.

<u>PROOF OF THE CLAIM</u>. We write

$$G^b(X,Y) = \partial_H\left(\begin{array}{c|c} S^b.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right)$$

and

$$G^b_d(X,Y) = \partial_H\left(\begin{array}{c|c} S^b_d.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right)$$

Terms like $G^b(X,Y)$ and $G^b_d(X,Y)$ can be rewritten using the Expansion Theorem. ET will yield $4 + 6 = 10$ terms and in all cases in this proof at most 2 of these terms are not equal to δ. In the sequel we will use applications of ET as a single calculation step. (Note that it is entirely feasible to verify all these applications of ET automatically.)

Now:

$$T^b(X,Y) = \sum_{d\in D} r1(d).\tau_I\partial_H\left(\begin{array}{c|c} S^b_d.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right) =$$

$$\sum_{d\in D} r1(d).\tau_I G^b_d(X,Y).$$

We will derive a recursive specification for $G^b_d(X,Y)$:

$$G^b_d(X,Y) = j.\partial_H\left(\begin{array}{c|c} U^b_d.S^{1-b}.X & (i.w4(e) + i.w4(db)).K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right) =$$

$$= j.\left[i.\partial_H\left(\begin{array}{c|c} U^b_d.S^{1-b}.X & w4(e).K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right) + i.\partial_H\left(\begin{array}{c|c} U^b_d.S^{1-b}.X & w4(db).K \\ \hline L & R^{1-b}.R^b.Y\end{array}\right)\right] =$$

$$= j.\left[i.j.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline L & w6(1-b).R^{1-b}.R^b.Y \end{array} \right) + \right.$$

$$\left. + \; i.j.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline L & w2(d).w6(b).R^b.Y \end{array} \right) \right] =$$

$$= j.\left[i.j.j.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline (i.w5(e) + i.w5(1-b)).L & R^{1-b}.R^b.Y \end{array} \right) + i.j.Z \right]$$

$$\text{(with } Z = \partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline L & w2(d).w6(b).R^b.Y \end{array} \right) \text{)} =$$

$$= j.\left[i.j.j.\left\{ i.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline w5(e).L & R^{1-b}.R^b.Y \end{array} \right) + \right.\right.$$

$$\left.\left. + \; i.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline w5(1-b).L & R^{1-b}.R^b.Y \end{array} \right) \right\} + i.j.Z \right] =$$

$$= j.\left[i.j.j.\left\{ i.j.\partial_H \left(\begin{array}{c|c} S_d^b.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y \end{array} \right) + \right.\right.$$

$$\left.\left. + \; i.j.\partial_H \left(\begin{array}{c|c} S_d^b.S^{1-b}.X & K \\ \hline L & R^{1-b}.R^b.Y \end{array} \right) \right\} + i.j.Z \right] =$$

$$= j.\left[i.j.j.i.j.G_d^b(X,Y) + i.j.Z \right].$$

We can now apply KFAR for $k = 6$ and $Y_0 = \delta$, $Y_1 = i.j.Z$, $Y_2 = \ldots = Y_5 = \delta$.
This gives:

$$\tau_I(G_d^b(X,Y)) = \tau.\tau_I(i.j.Z) \;.$$

Hence:

$$T^{b}(X,Y) = \sum_{d\in D} r1(d).\tau_I(G^{b}_{d}(X,Y)) = \sum_{d\in D} r1(d).\tau.\tau_I(i.j.Z) =$$

$$= \sum_{d\in D} r1(d).\tau.\tau.\tau.\tau_I(Z) = \sum_{d\in D} r1(d).\tau_I(Z) =$$

$$= \sum_{d\in D} r1(d).\tau_I\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline L & w2(d).w6(b).R^{b}.Y \end{array}\right) =$$

$$= \sum_{d\in D} r1(d).\tau_I\left[w2(d).\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline L & w6(b).R^{b}.Y \end{array}\right)\right] =$$

$$= \sum_{d\in D} r1(d).w2(d).\tau_I(K^{b}_{d}(X,Y))$$

$$\left(\text{with } K^{b}_{d}(X,Y) = \partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline L & w6(b).R^{b}.Y \end{array}\right)\right).$$

The next part of the proof of the claim consists in deriving a recursion equation for $K^{b}_{d}(X,Y)$:

$$K^{b}_{d}(X,Y) = j.\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline (i.w5(b)+i.w5(e)).L & R^{b}.Y \end{array}\right) =$$

$$= j.\left[i.\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline w5(b).L & R^{b}.Y \end{array}\right) + i.\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & K \\ \hline w5(e).L & R^{b}.Y \end{array}\right)\right] =$$

$$= j.\left[i.j.\partial_H\left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^{b}.Y \end{array}\right) + i.j.\partial_H\left(\begin{array}{c|c} S^{b}_{d}.S^{1-b}.X & K \\ \hline L & R^{b}.Y \end{array}\right)\right] =$$

$$= j.\left[i.j.V + i.j.j.\partial_H\left(\begin{array}{c|c} U^{b}_{d}.S^{1-b}.X & (i.w4(e)+i.w4(db)).K \\ \hline L & R^{b}.Y \end{array}\right)\right]$$

$$\text{(with } V = \partial_H \left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^b.Y \end{array}\right)\text{)} \quad =$$

$$= j.\left[i.j.V + i.j.j.\left\{ i.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & w4(e).K \\ \hline L & R^b.Y \end{array}\right) + \right.\right.$$

$$\left.\left. + \; i.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & w4(db).K \\ \hline L & R^b.Y \end{array}\right)\right\}\right] \quad =$$

$$= j.\left[i.j.V + i.j.j.\left\{ i.j.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline L & w6(b).R^b.Y \end{array}\right) + \right.\right.$$

$$\left.\left. + \; i.j.\partial_H \left(\begin{array}{c|c} U_d^b.S^{1-b}.X & K \\ \hline L & w6(b).R^b.Y \end{array}\right)\right\}\right] \quad =$$

$$= j.\left[i.j.V + i.j.j.i.j.K_d^b(X,Y)\right].$$

Applying KFAR we get:

$$\tau_I(K_d^b(X,Y)) = \tau.\tau_I(i.j.V) = \tau.\tau.\tau.\tau_I(V) = \tau.\tau_I(V) =$$

$$\tau.\tau_I\partial_H \left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^b.Y \end{array}\right).$$

We conclude:

$$T^b(X,Y) = \sum_{d \in D} r1(d).w2(d).\tau_I(K_d^b(X,Y)) =$$

$$= \sum_{d \in D} r1(d).w2(d).\tau.\tau_I\partial_H \left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^b.Y \end{array}\right) =$$

$$= \sum_{d\in D} r1(d).w2(d).\tau_I\partial_H\left(\begin{array}{c|c} S^{1-b}.X & K \\ \hline L & R^b.Y \end{array}\right).$$

This finishes the proof of the claim and the verification of ABP.

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] BARTLETT, K.A., R.A. SCANTLEBURY & P.T. WILKINSON, *A note on reliable full-duplex transmission over half duplex lines,* CACM 12,No.5(1969).

[3] BERGSTRA, J.A. & J.W. KLOP, *Process Algebra for Synchronous Communication,* Information and Control, Vol.60, Nos.1-3, 1984, 109-137.

[4] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes,* to appear in Proc. of the CWI Symposium Mathematics and Computer Science (eds. J.W. de Bakker, M. Hazewinkel and J.K. Lenstra), North-Holland, Amsterdam 1985.

[5] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes with abstraction,* to appear in Theoretical Computer Science, 1985.

[6] HAILPERN, B.T., *Verifying concurrent processes using temporal logic.* Springer LNCS 129, 1982.

[7] HAILPERN, B.T. & S. OWICKI, *Verifying network protocole using temporal logic,* in: Trends and applications symposium, National Bureau of Standards 1980.

[8] LAMPORT, L., *Specifying concurrent program modules,* ACM Toplas, Vol.5, No.2, p.190-222.

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

[10] SCHWARTZ, R.L. & P. MELIAR SMITH, *From state machine to temperal logic, specification methods for protocol standards,* IEEE Transactions on communication, Vol.30, No.12 (1982) p.2486-2496.

[11] YEMINI, Y. & J.F. KUROSE, *Can current protocol verification techniques guarantee correctness?* Computer networks, Vol.6, No.6 (1982), p. 377-381.