Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Bergstra, J.W. Klop

Verification of an alternating bit protocol
by means of process algebra

Department of Computer Science Report CS-R8404 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.).

VERIFICATION OF AN ALTERNATING BIT PROTOCOL BY MEANS OF PROCESS ALGEBRA

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

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.

1980 MATHEMATICS SUBJECT CLASSIFICATION: 68B10, 68C01, 68D25, 68F20. 69F11, 69F12, 69F32, 69F43
1982 CR. CATEGORIES: F.1.1, F.1.2, F.3.2, F.4.3.
KEY WORDS & PHRASES: process algebra, alternating bit protocol, abstraction, fair abstraction.
NOTE: This report will be submitted for publication elsewhere.

Report CS-R8404
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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:

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

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 simulatneously dealt with in the equational calculus of process algebra.

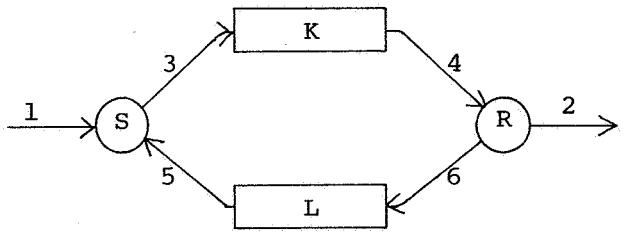
The structure of this note is as follows:

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

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:

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

Verification amounts to a proof that this T satisfies the equation

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

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

$$rt(a)|wt(a) = i,$$

where i is an internal action. Another kind of internal action is j. 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 $.|. : A \times A \rightarrow A$ yields δ (deadlock or failure) except in the case mentioned before: $rt(a)|wt(a) = i$.

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:

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

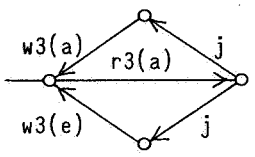
and I, the set of internal actions is just $\{i,j\}$.

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:

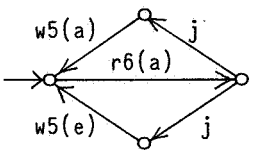
K:



$(a \in D0 \cup D1)$

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

L:



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

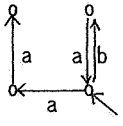
Note that K and L, after receiving input, have a nondeterministic choice, by doing one of both j actions.

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 j's.

Notation. We use the following diagram conventions:

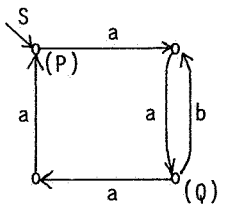
(1) A double labeled arrow $\xrightarrow{a\ b}$ stands for $\underset{b}{\overset{a}{\rightrightarrows}}$.

(2) If at a node s in graph g the notation P occurs then P is the process corresponding to the graph g with root at s. Thus in $\rightarrow \underset{b}{\overset{a}{\rightrightarrows}}_P$ we have "$P = \underset{b}{\overset{a}{\rightrightarrows}}$".

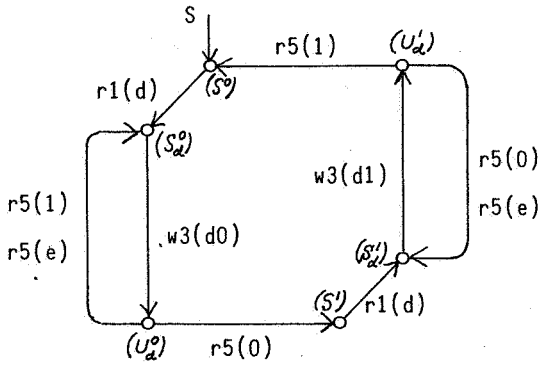
(3) If at nodes $s_1, s_2, \ldots$ in graph g the notations $(P_1), (P_2), \ldots$ (respectively) occur then P_i is the process corresponding to the graph g with root s_i and cut off whenever in the direction of the arrows a bracketed (P_j) is encountered. Thus in g as below we have $P = aa$, $Q =$ 

and $S = P\underline{Q}S = (PQ)^{\omega}$.

g:



Sender:



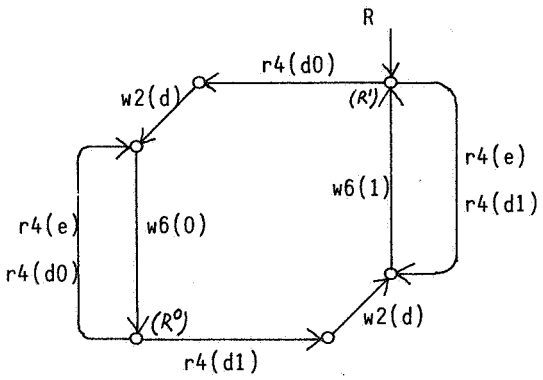
$$S = S^0.S^1.S$$

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

$$S_d^n = w3(dn).U_d^n$$

$$U_d^n = (r5(1-n) + r5(e)).S_d^n + r5(n)$$

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 $.|. : A \times A \rightarrow A$ 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 $\underline{X}$ for $\{X_i|i\in\omega\}$ and $\underline{Y}$ for $\{Y_i|i\in\omega\}$. 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 which is not in the scope of an operator τ_I with $I \cap E \neq \emptyset$. We illustrate this notion by means of some examples. Let $A = \{a,b,c,i\}$, $E = \{a,b,i\}$, $I = \{i\}$. Then the following terms are E-guarded resp. not E-guarded:

E-guarded	not E-guarded
$a.(X \| Y) + \tau_I(b.X)$	$\tau.X + b.Y$
$\tau.i.aX + \tau_I(i.aX))$	$\tau_I(\tau.i.X + a.Y)$
$a.X.(X \| Y) + \tau.(b \lfloor\!\lfloor X)$	$(a.X \| Y) \lfloor\!\lfloor b.Z$
$i.X + a.Y.\tau$	$\tau X \mid aY$

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

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 $S_E(P;P_i \mid i \in \omega)$ then we want to view $S_E(X;X_i \mid i \in \omega)$ 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:

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

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:

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

Here $\mathbb{Z}_k = \{0,\ldots,k-1\}$ 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. Handshaking axiom (HA).

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

$$(\text{HA}) \quad X|Y|Z = \delta$$

2.2.4. Expansion Theorem (ET).

This theorem, in the context of CCS due to MILNER [9] and for ACP_τ formulated in [4], can be shown for finite processes from ACP_τ. (See [5].) The Expansion Theorem presupposes HA. Let $X_1,\ldots,X_k$ be processes. With X^i we denote the merge of all X_n such that $n \in \{1,\ldots,k\}-\{i\}$. With $X^{i,j}$ we denote the merge of all X_n such that $n \in \{1,\ldots,k\}-\{i,j\}$.

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

$$\text{(ET)} \qquad X_1 \| \ldots \| X_k = \sum_{1 \leqslant i \leqslant k} X_i \between \gamma^i + \sum_{1 \leqslant i < j \leqslant k} (X_i | X_j) \between \gamma^{i,j}$$

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 $T^* = \sum_{d \in D} rl(d).w2(d).T^*$ 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:

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

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

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

CLAIM: $$T^b(X,Y) = \sum_{d \in D} rl(d).w2(d).\tau_I \partial_H \left(\frac{S^{1-b}.X \mid K}{L \mid R^b.Y} \right).$$

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

$$T = T^0(S,R) = \sum_{d \in D} rl(d).w2(d).\tau_I \partial_H \left(\frac{S^1.S \mid K}{L \mid R^0.R} \right) =$$

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

$$\sum_{d \in D} rl(d).w2(d). \sum_{a \in D} rl(a).w2(a).\tau_I \partial_H \left(\frac{S^0.S^1.S \mid K}{L \mid R^1.R^0.R} \right) =$$

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

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

PROOF OF THE CLAIM. We write

$$G^b(X,Y) = \partial_H \left(\frac{S^b.S^{1-b}.X \mid K}{L \mid R^{1-b}.R^b.Y} \right)$$

and

$$G^b_d(X,Y) = \partial_H \left(\frac{S^b_d.S^{1-b}.X \mid K}{L \mid R^{1-b}.R^b.Y} \right)$$

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:

$$T^b(X,Y) = \sum_{d \in D} rl(d).\tau_I \partial_H \left(\frac{S^b_d.S^{1-b}.X \mid K}{L \mid R^{1-b}.R^b.Y} \right) =$$

$$\sum_{d \in D} rl(d) . \tau_I G_d^b(X,Y) .$$

We will derive a recursive specification for $G_d^b(X,Y)$:

$$G_d^b(X,Y) = j.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{(i.w4(e) + i.w4(db)).K}{R^{1-b}.R^b.Y} \right) =$$

$$= j.\left[i.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{w4(e).K}{R^{1-b}.R^b.Y} \right) + i.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{w4(db).K}{R^{1-b}.R^b.Y} \right) \right] =$$

$$= j.\left[i.j.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{K}{w6(1-b).R^{1-b}.R^b.Y} \right) + \right.$$

$$\left. + i.j.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{K}{w2(d).w6(b).R^b.Y} \right) \right] =$$

$$= j.\left[i.j.j.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{(i.w5(e) + i.w5(1-b)).L} \,\middle|\, \frac{K}{R^{1-b}.R^b.Y} \right) + i.j.Z \right]$$

$$\text{(with } Z = \partial_H \left(\frac{U_d^b.S^{1-b}.X}{L} \,\middle|\, \frac{K}{w2(d).w6(b).R^b.Y} \right)) =$$

$$= j.\left[i.j.j.\left\{ i.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{w5(e).L} \,\middle|\, \frac{K}{R^{1-b}.R^b.Y} \right) + \right. \right.$$

$$\left. \left. + i.\partial_H \left(\frac{U_d^b.S^{1-b}.X}{w5(1-b).L} \,\middle|\, \frac{K}{R^{1-b}.R^b.Y} \right) \right\} + i.j.Z \right] =$$

$$= j.\left[i.j.j.\left\{i.j.\partial_H\left(\frac{S_d^b.S^{1-b}.X}{L}\,\middle|\,\frac{K}{R^{1-b}.R^b.Y}\right)+\right.\right.$$

$$\left.\left.+\,i.j.\partial_H\left(\frac{S_d^b.S^{1-b}.X}{L}\,\middle|\,\frac{K}{R^{1-b}.R^b.Y}\right)\right\}+\ i.j.Z\right]=$$

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

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:

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

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

$$= \sum_{d\in D} rl(d).\tau_I\partial_H\left(\frac{U_d^b.S^{1-b}.X}{L}\,\middle|\,\frac{K}{w2(d).w6(b).R^b.Y}\right)=$$

$$= \sum_{d\in D} rl(d).\tau_I\left[w2(d).\partial_H\left(\frac{U_d^b.S^{1-b}.X}{L}\,\middle|\,\frac{K}{w6(b).R^b.Y}\right)\right]=$$

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

$$(\text{with } K_d^b(X,Y) = \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid K}{L \mid w6(b) . R^b . Y} \right) .$$

The next part of the proof of the claim consists in deriving a recursion equation for $K_d^b(X,Y)$:

$$K_d^b(X,Y) = j . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid K}{(i.w5(b) + i.w5(e)).L \mid R^b . Y} \right) =$$

$$= j . \left[i . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid K}{w5(b) . L \mid R^b . Y} \right) + i . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid K}{w5(e) . L \mid R^b . Y} \right) \right] =$$

$$= j . \left[i . j . \partial_H \left(\frac{S^{1-b} . X \mid K}{L \mid R^b . Y} \right) + i . j . \partial_H \left(\frac{S_d^b . S^{1-b} . X \mid K}{L \mid R^b . Y} \right) \right] =$$

$$= j . \left[i . j . V + i . j . j . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid (i.w4(e) + i.w4(db)).K}{L \mid R^b . Y} \right) \right]$$

$$(\text{with } V = \partial_H \left(\frac{S^{1-b} . X \mid K}{L \mid R^b . Y} \right)) =$$

$$= j . \left[i . j . V + i . j . j . \left\{ i . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid w4(e) . K}{L \mid R^b . Y} \right) + \right. \right.$$

$$\left. \left. + i . \partial_H \left(\frac{U_d^b . S^{1-b} . X \mid w4(db) . K}{L \mid R^b . Y} \right) \right\} \right] =$$

$$= j.\left[i.j.V + i.j.j.\left\{i.j.\partial_H\left(\frac{U_d^b.S^{1-b}.X \mid K}{L \mid w6(b).R^b.Y}\right) + \right.\right.$$

$$\left.\left. + i.j.\partial_H\left(\frac{U_d^b.S^{1-b}.X \mid K}{L \mid w6(b).R^b.Y}\right)\right\}\right] =$$

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

Applying KFAR we get:

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

$$\tau.\tau_I\partial_H\left(\frac{S^{1-b}.X \mid K}{L \mid R^b.Y}\right).$$

We conclude:

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

$$= \sum_{d\in D} rl(d).w2(d).\tau.\tau_I\partial_H\left(\frac{S^{1-b}.X \mid K}{L \mid R^b.Y}\right) =$$

$$= \sum_{d\in D} rl(d).w2(d).\tau_I\partial_H\left(\frac{S^{1-b}.X \mid K}{L \mid R^b.Y}\right).$$

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 communicating and mutual exclusion*, Report IW 218/83, Mathematisch Centrum Amsterdam 1983.

[4] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes*, Report IW 2../84, Centrum voor Wiskunde en Informatica, Amsterdam 1984.

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

[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 quarantee correctness?* Computer networks, Vol.6, No.6 (1982), p. 377-381.