Two Simple Protocols

Frits W. Vaandrager
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

After some introductory remarks about the specification and verification of distributed systems in the framework of process algebra, simple versions of the Alternating Bit Protocol and the Positive Acknowledgement with Retransmission protocol are discussed.

1. General introduction

In the ACP formalism we can define (specify) networks of processes which cooperate in an *asynchronous* way. We can do this by looking at the communication channels in the network as processes which communicate in a *synchronous* way with the processors to which they are connected. Almost always, this synchronous communication will take place according to the *handshaking paradigm*: exactly two processes participate in every communication. When we specify communications of this type we will employ a *read/send* communication function: Let $\mathbb{D}$ be a finite set of *data* which can be communicated between processes, and let $\mathbb{P}$ be a finite set of *locations* (or *ports*) where synchronous communication can take place. The alphabet of atomic actions now consists of *read actions* $rp(d)$, *send actions* $sp(d)$ and *communication actions* $cp(d)$ for $p \in \mathbb{P}$ and $d \in \mathbb{D}$. As the only communications we have: $\gamma(rp(d), sp(d)) = cp(d)$.

A typical system that can be specified in this way in ACP is depicted in Figure 1. This graphical representation was first used by Jan Willem Klop. The corresponding process expression is then for instance:

$$\partial_H(P_1 \| P_2 \| P_3 \| P_4 \| P_5 \| C_1 \| C_2 \| C_3 \| C_4 \| C_5).$$

Let us stand still for a moment at the issue of the physical interpretation of expressions of this type and the question about the nature of the events in reality that are modelled by the read and send actions. In general we will describe with expressions P_1, P_2,... and C_1, C_2,... the behaviour of physical

Partial support received from the European Community under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

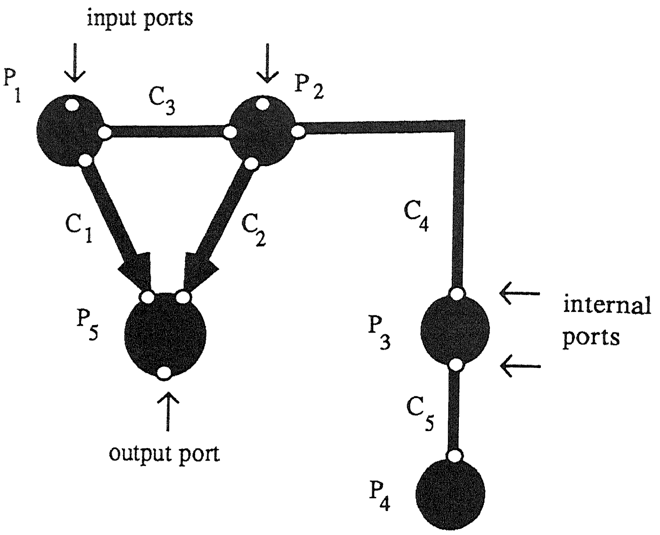

FIGURE 1

objects. P_1 and P_2 for example correspond with personal computers, P_3 and P_4 with disk drives and P_5 with a printer. C_1 up to C_5 describe cables of a network connecting all these machines together. All the components have a spatial extent. Now we associate with each port name $p \in \mathbb{P}$ a point in space on the border line between two (or more) components (see Figure 2).

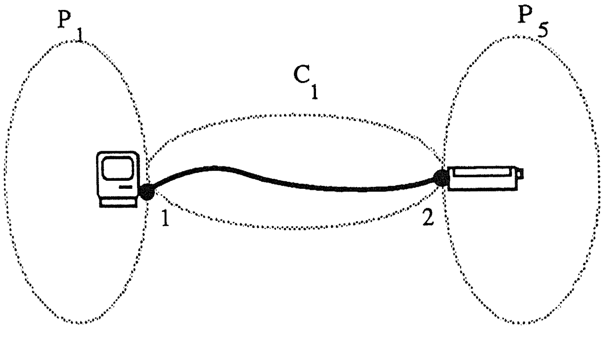

FIGURE 2

When process P_1 performs an action $s\,1(d_0)$ we relate this to the transmission of a datum d_0 by the personal computer. At the physical level this means that at the location (port 1) where cable C_1 is connected to the computer variations occur in the electric voltage during a certain amount of time. Because d_0 can have a considerable size (think of a file which is sent to the printer) the transmission can take a lot of time. The instantaneous event associated with $r\,1(d_0)$ occurs at a moment the cable 'knows' that a datum d_0 has been transmitted at port 1. Such a moment occurs when P_1 has almost finished transmission of this datum. The complementary event $s\,1(d_0)$ happens at the moment that the computer has made so much progress with the transmission of d_0 that the environment has enough information to 'know' that it is d_0 indeed. By defining the events in the right way we can ensure that $s\,1(d_0)$ and $r\,1(d_0)$ coincide. It is impossible that $s\,1(d_0)$ occurs and $r\,1(d_0)$ does not, or the other way around. Therefore we can consider the occurrence of $s\,1(d_0)$ and $r\,1(d_0)$ as a single event. This is precisely what we express in process algebra with the communication function and the encapsulation operator. Notice that the above interpretation of read and send actions is not in conflict with the intuition presented in [6] that the instantaneous event associated with an atomic process should be situated at the beginning of that process. Apparently a command **print**(d_0) that one can give to the computer corresponds to a process $\tau{\cdot}s\,1(d_0)$. At the moment process C_1 knows that d_0 is transmitted and the event $c\,1(d_0)$ occurs, the execution of processes $s\,1(d_0)$ and $r\,1(d_0)$ will not yet be finished. One possible scenario is that execution of $s\,1(d_0)$ finishes before the end of the execution of $r\,1(d_0)$.

In process theory we assume that the only thing which is interesting about a system is its external behaviour. Two systems with identical external behaviour should be identified in principle. From the point of view of process algebra there is no difference between a labourer assembling bicycle pumps, and a robot performing the same job. Unless attention is paid in the formal specification to all kind of details like fluctuations in productivity due to nocturnal excesses, the approaching weekend, depressions because of the monotony of the job, etc.

In order to realise a certain external behaviour (the *specification*), often a complex internal structure (the *implementation*) is needed. This brings us to the important issue of *abstraction*. We are interested in a technique which makes it possible to *abstract* from the internal structure of a system, so that we can derive statements about the external behaviour. Abstraction is an indispensable tool for managing the complexity of process verifications. This is because abstraction allows for a reduction of the complexity (the number of states) of subprocesses. This makes it possible to verify large processes in a *hierarchical* way. A typical verification consists of a proof that, after abstraction, an implementation *IMP* behaves like the much simpler process *SPEC* which serves as system specification:

$$ABS(IMP) = SPEC.$$

In process algebra we model abstraction by making the distinction between two types of actions, namely *external* or *observable* actions and the *internal* or *hidden* action τ, and by introducing explicit abstraction operators τ_I which transform observable actions into the hidden action (see Figure 3).

Fundamental within the ACP-formalism is the *algebraic* approach. A verification consists of a proof of a statement of the form:

$$ACP_\tau + \cdots \vdash \tau_I(IMP) = SPEC.$$

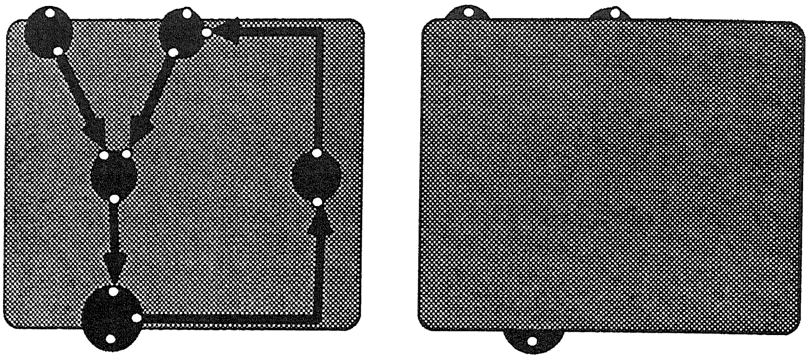

FIGURE 3

The idea is that 'users' can stay in the realm of the formal system and execute algebraic manipulations, without the need for an excursion into the semantics.

2. THE ALTERNATING BIT PROTOCOL

The most studied communication protocol in existence is undoubtedly the Alternating Bit Protocol (ABP). Whenever somewhere in this world someone introduces a new formalism for concurrent processes, you can count on it that the practical applicability of the formalism is illustrated by means of a specification and verification of a variant of the ABP. As a first test-case for a concurrency theory the protocol is very appropriate indeed: the protocol can be described in a few words, but the formal specification and verification of it forms a non-trivial problem. However, for real practical application of a concurrency theory much more is needed. In the analysis of realistic protocols one encounters various problems of scale which cannot be observed when dealing with the ABP.

We do not want to break with the traditions concerning the ABP, and will start here with a discussion of a simple variant of the ABP in the setting of process algebra. More complex protocols are dealt with in some other contributions of this volume.

Other discussions of the Alternating Bit Protocol can be found in [2,8,10,12,15]. In the context of ACP the protocol was verified for the first time in [4]. The discussion of the ABP here is based on a streamlined version of the proof, given by the author, which can be found in [5]. Variants of the ABP are discussed in the setting of process algebra in [7,9].

2.1. Specification

The Alternating Bit Protocol can be visualised as follows:

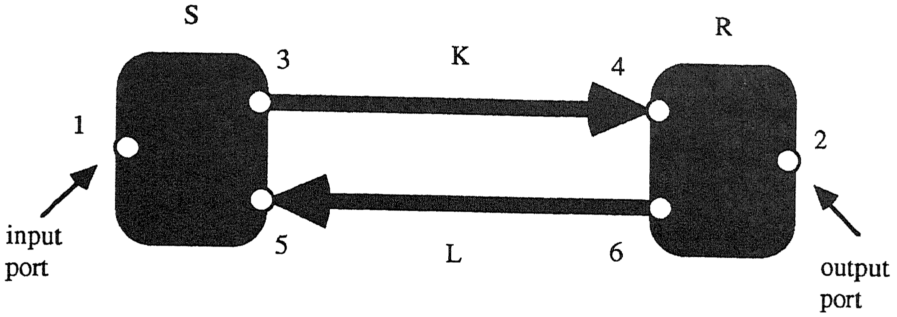

FIGURE 4

Let D be a finite set of data. Elements of D are to be transmitted by the protocol from port 1 to port 2. There are four components: a sender S, a receiver R, and two channels K and L.

2.1.1. Component S. S starts by Reading a Message (RM) at port 1. Then a *frame* consisting of the message from D and a control bit is transmitted via channel K (SF = Send Frame), until a correct acknowledgement has arrived via channel L (RA = Receive Acknowledgement). In equations we will always use the symbol d to denote elements from the set D, b denotes an element from $B = \{0,1\}$, and f finally is used for frames in $D \times B$. In Table 1 we 'declare' the recursive specification that gives the behaviour of component S. After a variable has been declared we will use it without mentioning the corresponding specification.

$$\begin{aligned}
S &= RM^0 \\
RM^b &= \sum_{d \in D} r1(d) \cdot SF^{db} \\
SF^{db} &= s3(db) \cdot RA^{db} \\
RA^{db} &= (r5(1-b) + r5(ce)) \cdot SF^{db} + r5(b) \cdot RM^{1-b}
\end{aligned}$$

TABLE 1. Recursive specification for component S

Graphically we can depict process S as in Figure 5. In a certain sense the figure is inaccurate: instead of a node SF^{e0} for each element e in D, there is only a single node SF^{d0}. Between each pair of nodes we draw only one edge, which however can be labelled with more than one action. Figure 5 can be considered as a 'projection' of the transition diagram belonging to S.

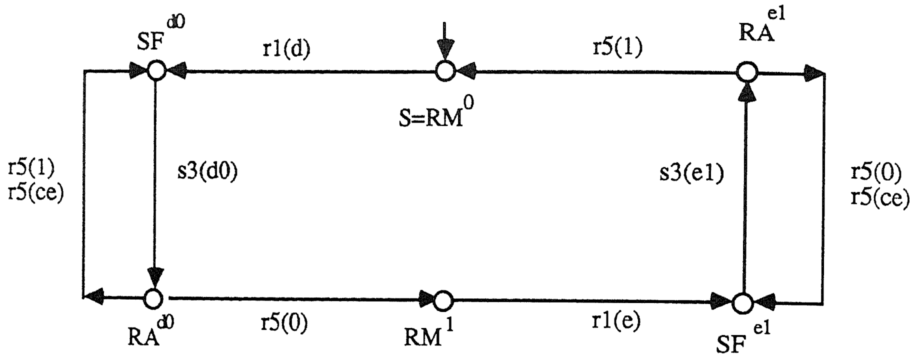

FIGURE 5

2.1.2. Component K. We assume that two things can happen if we send a frame into channel K: (1) the message is communicated correctly, (2) the message is damaged in transit. We assume that if something goes wrong with the message, the receiver hardware will detect this when it computes a *checksum* (ce = checksum error). Further the channels are supposed to be fair in the sense that they will not produce an infinite consecutive sequence of erroneous outputs. These are plausible assumptions we have to make in order to prove correctness of a protocol that is based on unreliable message passing. Data transmission channel K communicates frames in the set $D \times B$ from port 3 to 4. We give the defining equations (Table 2) and the corresponding diagram (Figure 6).

$$K = \sum_{f \in D \times B} r3(f) \cdot K^f$$

$$K^f = (\tau \cdot s4(ce) + \tau \cdot s4(f)) \cdot K$$

TABLE 2. Defining equations for channel K

The τ's in the second equation express that the choice whether or not a frame f is to be communicated correctly, is nondeterministic and cannot be influenced by one of the other components.

2.1.3. Component R. R starts by Receiving a Frame (RF) via channel K. If the control bit of the frame is correct, then the message contained in the frame is sent to port 2 (SM = Send Message). Component R Sends Acknowledgements (SA) via channel L. Figure 7 gives the transition diagram for R.

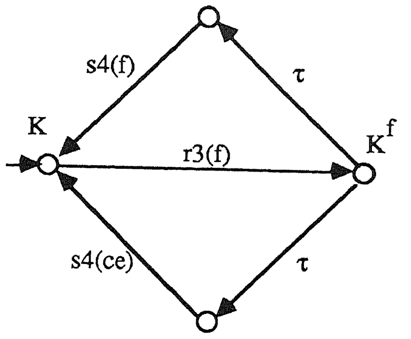

FIGURE 6

$$
\begin{aligned}
R &= RF^0\\
RF^b &= \Big(\sum_{d\in D} r4(d(1-b)) + r4(ce)\Big)\cdot SA^{1-b} + \sum_{d\in D} r4(db)\cdot SM^{db}\\
SA^b &= s6(b)\cdot RF^{1-b}\\
SM^{db} &= s2(d)\cdot SA^b
\end{aligned}
$$

TABLE 3. Recursive specification for component R

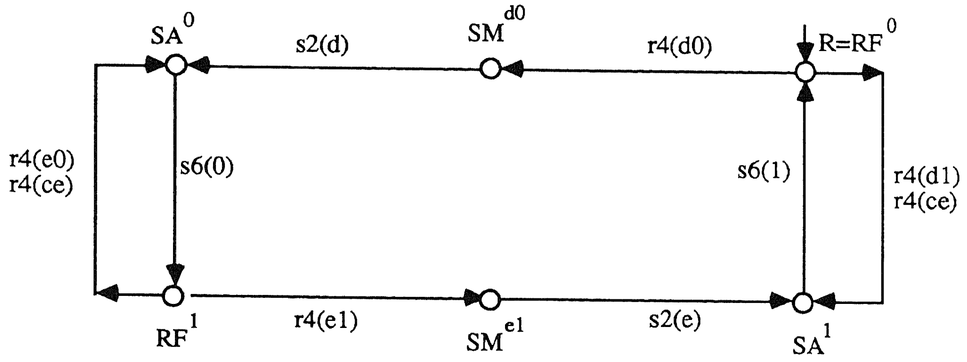

FIGURE 7

2.1.4. Component L. The task of acknowledgement transmission channel L is to communicate boolean values from R to S. The channel may yield error outputs but again we assume that this is detected, and that moreover the channel is fair. See Figure 8 for the diagram.

$$\boxed{\begin{aligned} L &= \sum_{b\in B} r6(b)\cdot L^b \\ L^b &= (\tau\cdot s5(ce)+\tau\cdot s5(b))\cdot L \end{aligned}}$$

TABLE 4. Defining equations for channel L

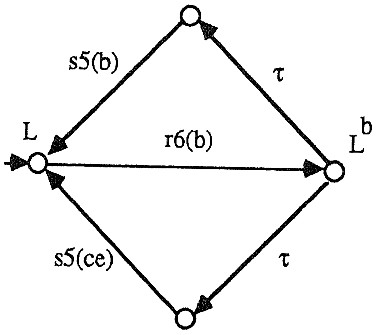

FIGURE 8

2.1.5. Sets. Define $\mathbb{D}=D\cup(D\times B)\cup B\cup\{ce\}$. $\mathbb{D}$ is the set of 'generalised' data (i.e. plain data, frames, bits, error) that occur as parameter of atomic actions. We use the notation $g\in\mathbb{D}$. The second parameter of atomic actions is the set $\mathbb{P}=\{1,2,\ldots,6\}$ of ports. We use symbol p for elements of $\mathbb{P}$. Communication follows the read/send scheme. This leads to an alphabet

$$A=\{sp(g),\ rp(g),\ cp(g)\,|\,p\in\mathbb{P}, g\in\mathbb{D}\}$$

and communications $\gamma(sp(g),\ rp(g))=cp(g)$ voor $p\in\mathbb{P}$, $g\in\mathbb{D}$. Define the following two subsets of A:

$$H=\{sp(g),\ rp(g)\,|\,p\in\{3,4,5,6\}, g\in\mathbb{D}\},$$
$$I=\{cp(g)\,|\,p\in\{3,4,5,6\}, g\in\mathbb{D}\}.$$

Now the ABP is described by

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

This is a good description in the sense that the specifications of the components S, K, R and L are guarded and consequently the specification of the ABP as a whole has a unique solution.

2.2. *Verification*

Verification of the ABP amounts to a proof that:

(1) the protocol will eventually send at port 2 all and only data it has read at port 1,

(2) the protocol will output data at port 2 in the same order as it has read them at port 1.

This means that, in order to verify the protocol, it is enough to prove the following theorem.

THEOREM 2.2.1. $ACP_\tau + SC + RDP + RSP + CA + CFAR \vdash$

$$ABP = \sum_{d \in D} r1(d) \cdot s2(d) \cdot ABP.$$

PROOF. Let $I' = \{cp(g) \mid p \in \{3,4,5\},\ g \in \mathbb{D}\}$. We will use $[x]$ as a notation for $\tau_{I'} \circ \partial_H(x)$. I' is defined in such a way that we just can derive a guarded system of equations for $[x]$. Consider the following system of recursion equations in Table 5.

(0) X	$=$	X_1^0
(1) X_1^b	$=$	$\sum_{d \in D} r1(d) \cdot X_2^{db}$
(2) X_2^{db}	$=$	$\tau \cdot X_3^{db} + \tau \cdot X_4^{db}$
(3) X_3^{db}	$=$	$c6(1-b) \cdot X_2^{db}$
(4) X_4^{db}	$=$	$s2(d) \cdot X_5^{db}$
(5) X_5^{db}	$=$	$c6(b) \cdot X_6^{db}$
(6) X_6^{db}	$=$	$\tau \cdot X_5^{db} + \tau \cdot X_1^{1-b}$

TABLE 5. Recursion equations for X

The transition diagram of X is displayed in Figure 9. We claim that with the above mentioned axioms one can prove that $X = [S \| K \| R \| L]$. We prove this by showing that $[S \| K \| R \| L]$ satisfies the same recursion equations (0)-(6) as X does. In the computations below, the bold-face part denotes the part of the expression currently being 'rewritten'.

$$[S \| K \| R \| L] = [RM^0 \| K \| RF^0 \| L] \qquad (0)$$

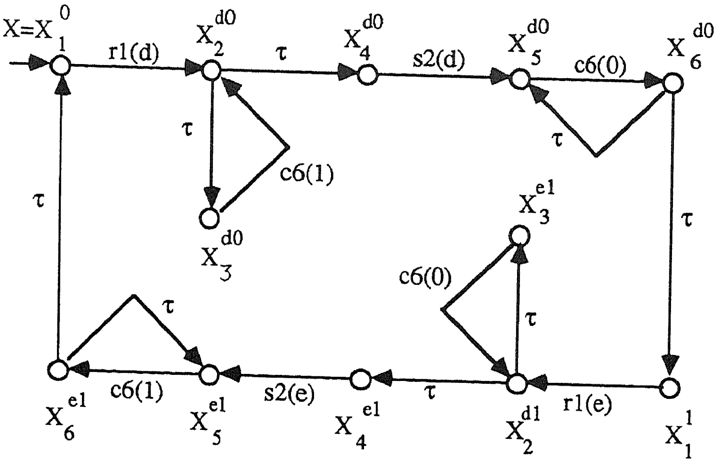

FIGURE 9

$$\begin{aligned}[\mathbf{RM^b}\|K\|RF^b\|L] &= \sum_{d\in D} r1(d)\cdot[\mathbf{SF^{db}}\|\mathbf{K}\|RF^b\|L]\\ &= \sum_{d\in D} r1(d)\cdot\boldsymbol{\tau}\cdot[RA^{db}\|K^{db}\|RF^b\|L]\\ &= \sum_{d\in D} r1(d)\cdot[RA^{db}\|K^{db}\|RF^b\|L]\end{aligned}\tag{1}$$

$$\begin{aligned}[RA^{db}\|\mathbf{K^{db}}\|RF^b\|L] &= \tau\cdot[RA^{db}\|\mathbf{s4(ce)\cdot K}\|\mathbf{RF^b}\|L] +\\ &\quad + \tau\cdot[RA^{db}\|\mathbf{s4(db)\cdot K}\|\mathbf{RF^b}\|L]\\ &= \tau\cdot[RA^{db}\|K\|SA^{1-b}\|L] + \tau\cdot[RA^{db}\|K\|SM^{db}\|L]\end{aligned}\tag{2}$$

$$\begin{aligned}[RA^{db}\|K\|\mathbf{SA^{1-b}}\|\mathbf{L}] &= c6(1-b)\cdot[RA^{db}\|K\|RF^b\|\mathbf{L^{1-b}}]\\ &= c6(1-b)\cdot(\tau\cdot[\mathbf{RA^{db}}\|K\|RF^b\|\mathbf{s5(ce)\cdot L}] +\\ &\quad + \tau\cdot[\mathbf{RA^{db}}\|K\|RF^b\|\mathbf{s5(1-b)\cdot L}])\\ &= c6(1-b)\cdot\tau\cdot\tau\cdot[\mathbf{SF^{db}}\|\mathbf{K}\|RF^b\|L]\\ &= c6(1-b)\cdot\boldsymbol{\tau}\cdot\boldsymbol{\tau}\cdot\boldsymbol{\tau}\cdot[RA^{db}\|K^{db}\|RF^b\|L]\\ &= c6(1-b)\cdot[RA^{db}\|K^{db}\|RF^b\|L]\end{aligned}\tag{3}$$

$$[RA^{db}\|K\|\mathbf{SM^{db}}\|L] = s2(d)\cdot[RA^{db}\|K\|SA^b\|L]\tag{4}$$

$$[RA^{db}\|K\|\mathbf{SA^b}\|\mathbf{L}] = c6(b)\cdot[RA^{db}\|K\|RF^{1-b}\|L^b]\tag{5}$$

$$\begin{aligned}[RA^{db}\|K\|RF^{1-b}\|\mathbf{L^b}] &= \tau\cdot[\mathbf{RA^{db}}\|K\|RF^{1-b}\|\mathbf{s5(ce)\cdot L}] +\\ &\quad + \tau\cdot[\mathbf{RA^{db}}\|K\|RF^{1-b}\|\mathbf{s5(b)\cdot L}]\\ &= \tau\cdot[SF^{db}\|K\|RF^{1-b}\|L] + \tau\cdot[RM^{1-b}\|K\|RF^{1-b}\|L]\end{aligned}\tag{6}$$

$$\begin{aligned}[\mathbf{SF^{db}}\|\mathbf{K}\|RF^{1-b}\|L] &= \tau\cdot[RA^{db}\|\mathbf{K^{db}}\|RF^{1-b}\|L] \\ &= \tau\cdot(\tau\cdot[RA^{db}\|\mathbf{s4(ce)}\cdot\mathbf{K}\|\mathbf{RF^{1-b}}\|L] + \\ &\quad + \tau\cdot[RA^{db}\|\mathbf{s4(db)}\cdot\mathbf{K}\|\mathbf{RF^{1-b}}\|L]) \\ &= \tau\cdot[RA^{db}\|K\|SA^{b}\|L]\end{aligned} \tag{7}$$

Now substitute (7) in (6) and apply RSP. Using conditional axiom CA6 we have $ABP = \tau_I([S\|K\|R\|L]) = \tau_I(X) = \tau_I(X_1^0)$. Further, an application of CFAR gives $\tau_I(X_2^{db}) = \tau\cdot\tau_I(X_4^{db})$ and $\tau_I(X_5^{db}) = \tau\cdot\tau_I(X_1^{1-b})$. Hence,

$$\begin{aligned}\tau_I(X_1^b) &= \sum_{d\in D} r1(d)\cdot\tau_I(X_2^{db}) = \sum_{d\in D} r1(d)\cdot\tau_I(X_4^{db}) \\ &= \sum_{d\in D} r1(d)\cdot s2(d)\cdot\tau_I(X_5^{db}) = \sum_{d\in D} r1(d)\cdot s2(d)\cdot\tau_I(X_1^{1-b})\end{aligned}$$

and thus

$$\tau_I(X_1^0) = \sum_{d\in D} r1(d)\cdot s2(d)\cdot\sum_{e\in D} r1(e)\cdot s2(e)\cdot\tau_I(X_1^0) \quad \text{and}$$

$$\tau_I(X_1^1) = \sum_{d\in D} r1(d)\cdot s2(d)\cdot\sum_{e\in D} r1(e)\cdot s2(e)\cdot\tau_I(X_1^1).$$

Applying RSP again yields $\tau_I(X_1^0) = \tau_I(X_1^1)$ and therefore

$$\tau_I(X_1^0) = \sum_{d\in D} r1(d)\cdot s2(d)\cdot\tau_I(X_1^0).$$

This finishes the proof of the theorem. □

REMARK. Channels K and L can contain only one datum at a time. Now one can say that this is no problem because S and R will never send a message into a channel when the previous one is still there. If S and R would do this then our process algebra modelling would be incorrect. Because they don't, there is no problem. This argument is correct for the ABP, but one should be careful in more complex situations: if one *implicitly* uses assumptions about the behaviour of a system in the specification of that system, then there is a danger that a verification shows that the system has a lot of 'wonderful' properties which in reality it has not. We give an example. Consider the situation where a process S first sends three threatening letters into channel K followed by an violent attempt to eliminate process R. Suppose K is a 1-datum-buffer. The system starts and S sends the first threatening letter into the channel. Now receiver R at the other side of the channel is very busy doing other things, and has no time to read messages from K. Only after a long, long time R looks if there is mail in K. Of course R is really shocked by the contents of the letter, and immediately tries to eliminate S. Only after this has succeeded, it reads from K again. Because S becomes dangerous only after the third message has been sent, process R will not get into trouble. The crucial point is now that this would have been different if K were a FIFO-queue.

3. The PAR protocol (part 1)

In this section we will describe a protocol that is very similar to the ABP, although there is a fundamental difference. The protocol, that is described in [13], is called PAR, which stands for *Positive Acknowledgement with Retransmission*. In the protocol the sender waits for an acknowledgement before a new datum is transmitted. Instead of two different acknowledgements, like in the ABP, the PAR protocol only uses one type of acknowledgement (hence the word 'Positive'). This discussion of the PAR protocol is a revised version of Sections 3 and 4 of [14].

3.1. *Specification*

The diagram that describes the architecture of the PAR protocol is identical to the diagram for the ABP, with as only difference that on one side of the sender a small *timer process* has been added.

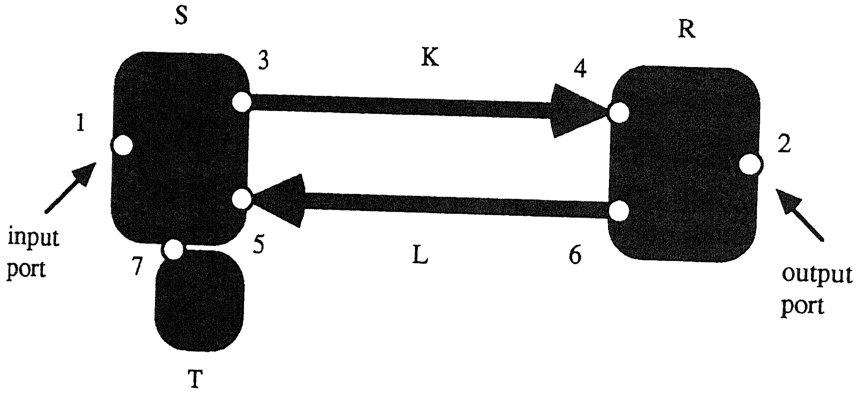

Figure 10

Thus, there are five components:
S: Sender
T: Timer
K: Data transmission channel
R: Receiver
L: Acknowledgement transmission channel

3.1.1. Sets. Let D be a finite set of data. Elements of D are to be transmitted by the PAR protocol from port 1 to port 2. Let $B=\{0,1\}$. Frames in $D\times B$ are transmitted by channel K. Define $\mathbb{D} = D\cup(D\times B)\cup\{ac,ce,st,to\}$ (ac =acknowledgement, ce=checksum error, st=start timer, to=time out). For the interaction with their environment, the components use ports from a set $\mathbb{P} = \{1,2,...,6,7\}$. $\mathbb{P}$ and $\mathbb{D}$ occur as parameters of atomic actions. Alphabet A and communication function γ are defined using the read/send scheme. In addition, A contains two other actions i and j which do not communicate.

3.1.2. The channels. If a message is sent into channel K or L, three things can happen:

(1) the message is communicated correctly,
(2) the message is damaged in transit,
(3) the message gets lost in the channel.

Channels K and L are described by the equations in Table 6.

$$\begin{array}{|lcl|}\hline K & = & \sum_{f \in D \times B} r3(f) \cdot K^f \\ K^f & = & (i \cdot s4(f) + i \cdot s4(ce) + i) \cdot K \\ \hline L & = & r6(ac) \cdot L^{ac} \\ L^{ac} & = & (j \cdot s5(ac) + j \cdot s5(ce) + j) \cdot L \\ \hline \end{array}$$

TABLE 6. Definition for channels K and L

The reason why we use actions i and j, instead of the τ as is done in the pecification of the ABP, will become clear further on.

.1.3. The sender. In the specification of the sender process S (Table 7) we use ormal variables RH^n, SF^{dn}, ST^{dn}, WS^{dn} ($d \in D$, $n \in B$):

H = Read a message from the Host at port 1. The host process, which is not specified here, furnishes the sender with data.
F = Send a Frame in channel K at port 3.
T = Start the Timer.
VS = Wait for Something to happen. Here there are three possibilities: (1) an acknowledgement frame arrives undamaged, (2) something damaged comes in, or (3) the timer goes off. If a valid acknowledgement comes in, the sender fetches the next message, and changes the control bit, otherwise a duplicate of the old frame is sent.

1.4. The timer. The timer process T is very simple (see Table 8). There are vo states: the initial (stop-) state and the (run-) state in which the timer is unning. In both states the timer can be started, but only in the running state time out can be generated.

$$
\begin{aligned}
S &= RH^0\\
RH^n &= \sum_{d\in D} r1(d)\cdot SF^{dn}\\
SF^{dn} &= s3(dn)\cdot ST^{dn}\\
ST^{dn} &= s7(st)\cdot WS^{dn}\\
WS^{dn} &= r5(ac)\cdot RH^{1-n}+(r5(ce)+r7(to))\cdot SF^{dn}
\end{aligned}
$$

TABLE 7. Specification of the sender process S

$$
\begin{aligned}
T &= r7(st)\cdot T^r\\
T^r &= r7(st)\cdot T^r+s7(to)\cdot T
\end{aligned}
$$

TABLE 8. Specification of the timer process T

3.1.5. The receiver. For the specification of the receiver process R (see Table 9) we use formal variables WF^n, SA^n, SH^{dn} $(d\in D,\ n\in B)$:

$WF =$ Wait for the arrival of a Frame at port 4.
$SA =$ Send an Acknowledgement at port 6.
$SH =$ Send a message to the Host at port 2. In general the host of the receiver will of course be different than the host of the sender.

$$
\begin{aligned}
R &= WF^0\\
WF^n &= r4(ce)\cdot WF^n+\sum_{d\in D} r4(d(1-n))\cdot SA^n+\sum_{d\in D} r4(dn)\cdot SH^{dn}\\
SA^n &= s6(ac)\cdot WF^n\\
SH^{dn} &= s2(d)\cdot SA^{1-n}
\end{aligned}
$$

TABLE 9. Specification of the receiver process R

When a valid frame arrives at the receiver, its control bit is checked to see if it is a duplicate. If not, it is accepted, the message contained in it is written at port 2, and an acknowledgement is generated. Duplicates and damaged frames are not written at port 2.

3.1.6. Premature time outs. We define

$$H = \{sp(g),\ rp(g) \mid p \in \{3,4,5,6,7\},\ g \in \mathbb{D}\}$$

and consider the expression

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

Each time after a frame is sent, the sender S starts the timer. An unpleasant property of the PAR protocol is that a premature time out can disturb the functioning of the protocol. If the sender times out too early, while the acknowledgement is still on the way, it will send a duplicate. When the previous acknowledgement finally arrives, the sender will mistakenly think that the just sent frame is the one being acknowledged and will not realise that there is potentially another acknowledgement somewhere in the channel. If the next frame sent is lost completely, but the extra acknowledgement arrives correctly, the sender will not attempt to retransmit the lost frame, and the protocol will fail.

An important observation is that in our modelling 'too early' corresponds exactly to the availability of an alternative action. Thus we can express the desired behaviour of the timer by giving the action $c7(to)$ a *lower priority* then every other atomic action. In the next section we will elaborate on this idea.

4. Priorities

The axiom system ACP$_\theta$, introduced in [1], consists of the operators and axioms of ACP, extended with a unary *priority* operator θ, an auxiliary binary operator $\lhd$ (*unless*) and some defining axioms for these operators. We use θ to model priorities. Parameter of θ is a partial order $<$ on the atomic actions. So for $a,b,c \in A$ we have

1. $\neg(a<a)$
2. $a<b \,\&\, b<c \Rightarrow a<c.$

The constant δ can be incorporated in this ordering as a minimal element. We then have $\delta<a$ for all $a \in A$. Consider, as an example, the following partial order on atomic actions a,b and c:

$$b<a \quad \text{and} \quad c<a.$$

Relative to this ordering the operator θ will forbid in a sum-context all actions that are majorated by one of the other actions in that sum-context. So we have for instance:

(i) $\theta(a+b)=a,\ \theta(a+c)=a$ but
(ii) $\theta(b+c)=b+c.$

Operator θ is axiomatized in the system ACP$_\theta$ (see Table 10).

Example. Let $b<a$ and $c<a$. Then:

(i) $\theta(a+b)=\theta(a)\lhd b+\theta(b)\lhd a = a\lhd b+b\lhd a = a+\delta = a,$
(ii) $\theta(b+c)=\theta(b)\lhd c+\theta(c)\lhd b = b\lhd c+c\lhd b = b+c,$

(iii) $\theta(b(a+c)) = \theta(b)\cdot\theta(a+c) = \ b\cdot(\theta(a)\lhd c + \theta(c)\lhd a) = \ b(a\lhd c + c\lhd a)$
$= b(a+\delta) = ba.$

ACP_θ			
$x+y = y+x$	A1	$a\lhd b = a \quad$ if $\neg(a<b)$	P1
$x+(y+z) = (x+y)+z$	A2	$a\lhd b = \delta \quad$ if $a<b$	P2
$x+x = x$	A3	$x\lhd yz = x\lhd y$	P3
$(x+y)z = xz+yz$	A4	$x\lhd(y+z) = (x\lhd y)\lhd z$	P4
$(xy)z = x(yz)$	A5	$xy\lhd z = (x\lhd z)y$	P5
$x+\delta = x$	A6	$(x+y)\lhd z = x\lhd z+y\lhd z$	P6
$\delta x = \delta$	A7		
$a\mid b = \gamma(a,b)$	CF		
$x\|y = x\mathbin{\lfloor\!\lfloor} y+y\mathbin{\lfloor\!\lfloor} x+x\mid y$	CM1	$\theta(a) = a$	TH1
$a\mathbin{\lfloor\!\lfloor} x = ax$	CM2	$\theta(xy) = \theta(x)\cdot\theta(y)$	TH2
$ax\mathbin{\lfloor\!\lfloor} y = a(x\|y)$	CM3	$\theta(x+y) = \theta(x)\lhd y+\theta(y)\lhd x$	TH3
$(x+y)\mathbin{\lfloor\!\lfloor} z = x\mathbin{\lfloor\!\lfloor} z+y\mathbin{\lfloor\!\lfloor} z$	CM4		
$(ax)\mid b = (a\mid b)x$	CM5		
$a\mid(bx) = (a\mid b)x$	CM6		
$(ax)\mid(by) = (a\mid b)(x\|y)$	CM7		
$(x+y)\mid z = x\mid z+y\mid z$	CM8		
$x\mid(y+z) = x\mid y+x\mid z$	CM9		
$\partial_H(a) = a \quad$ if $a\notin H$	D1		
$\partial_H(a) = \delta \quad$ if $a\in H$	D2		
$\partial_H(x+y) = \partial_H(x)+\partial_H(y)$	D3		
$\partial_H(xy) = \partial_H(x)\cdot\partial_H(y)$	D4		

TABLE 10

In [1] the proof can be found of the following theorem:

THEOREM 4.1.

i) *For each recursion-free closed ACP_θ-term s there is a basic term t such that $ACP_\theta \vdash s = t$.*

ii) *ACP_θ is a conservative extension of ACP, i.e. for all recursion-free ACP-terms s,t we have: $ACP_\theta \vdash s = t \Rightarrow ACP \vdash s = t$.*

At present it is not altogether clear how ACP_θ and ACP_τ should be combined into $ACP_{\tau\theta}$. As a consequence of Theorem 4.1 however we can give meaning to a term like $\tau_I(s)$, where s is a recursion-free closed ACP_θ-term. Expression s is related to exactly one ACP process, and ACP_τ is a conservative extension of

ACP. For infinite processes the situation is a bit more complicated. Without proof we mention that for the theory of regular process expressions (expressions that generate a finite transition diagram) we also have conservativity.

5. THE PAR PROTOCOL (PART 2)

Returning to the specification of the PAR protocol we define operator θ on the basis of the following partial ordering $<$ on A:
(1) $a < c7(st)$ for $a \in A - \{c7(st)\}$
(2) $c7(to) < a$ for $a \in A - \{c7(to)\}$.
The reason for giving action $c7(to)$ a lower priority than the other actions has already been given in Section 3.1.6. In addition we have given action $c7(st)$ a higher priority than the other actions in order to express that immediately after sending a message the timer is started. This assumption is not essential for the correctness of the protocol. The system as a whole is now described by

$$\theta \circ \partial_H(S \| T \| K \| R \| L).$$

The fact that in the scope of a priority operator no τ's are allowed explains the use of i and j actions in the specification of components K and L. We are only interested in the actions taking place at ports 1 and 2. The other actions cannot be observed.

$$I = \{cp(g) \mid p \in \{3,4,5,6,7\},\ g \in \mathbb{D}\} \cup \{i,j\}$$

The PAR protocol can now be specified by:

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

For a verification of the protocol it is enough to prove the following theorem.

THEOREM 5.1. $ACP_\tau + ACP_\theta + SC + RDP + RSP + CA + CFAR \vdash$

$$PAR = \sum_{d \in D} r1(d) \cdot s2(d) \cdot PAR$$

PROOF. Let $I' = \{cp(g) \mid p \in \{4,5,7\}, g \in \mathbb{D}\} \cup \{i,j\}$. We use $[x]$ as notation for $\tau_{I'} \circ \theta \circ \partial_H(x)$. Since $I' \subseteq I$ we can apply axiom CA6:

$$PAR = \tau_I([S \| T \| K \| R \| L]).$$

In the first part of the proof we will derive a guarded system of recursion equations for the process expression $[S \| T \| K \| R \| L]$ in which only the operators $+$ and $\cdot$ occur. Thereafter, in the second part, we will abstract from the other internal actions using CFAR. Throughout the proof d ranges over D and n ranges over B. The transition diagram of $[S \| T \| K \| R \| L]$ is depicted in Figure 11.

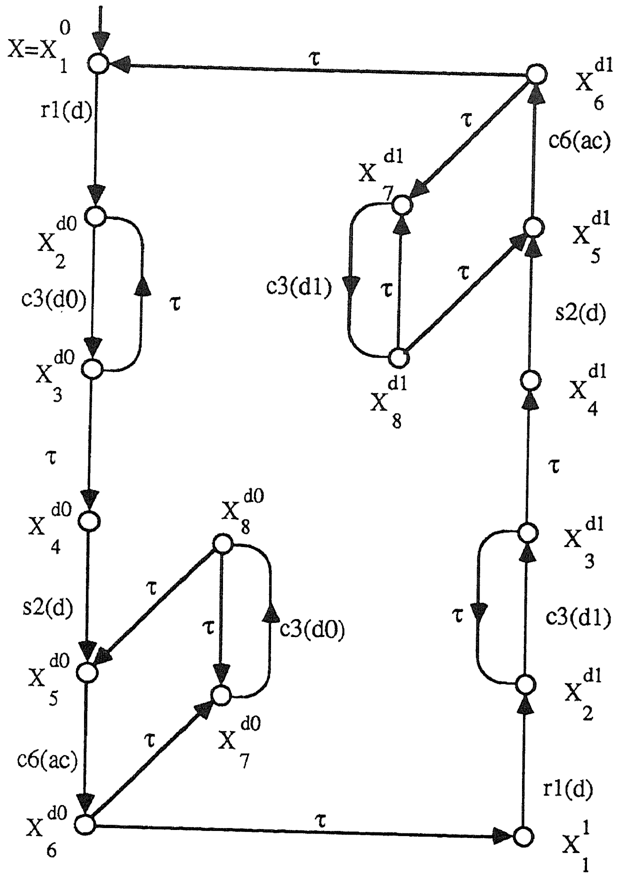

FIGURE 11

$$[S\|T\|K\|R\|L] = [RH^0\|T\|K\|WF^0\|L] \tag{0}$$

$$[RH^n\|T\|K\|WF^n\|L] = \sum_{d\in D} r1(d)\cdot[SF^{dn}\|T\|K\|WF^n\|L] \tag{1}$$

$$\begin{aligned}[SF^{dn}\|T\|K\|WF^n\|L] &= c3(dn)\cdot[ST^{dn}\|T\|K^{dn}\|WF^n\|L] \\ &= c3(dn)\cdot[WS^{dn}\|T^r\|K^{dn}\|WF^n\|L]\end{aligned} \tag{2}$$

(Here we used that the action $c7(st)$ has higher priority than the other actions.)

$$\begin{aligned}[WS^{dn}\|T^r\|K^{dn}\|WF^n\|L] = \tau_{I'}\circ\theta(&c7(to)\cdot\partial_H(SF^{dn}\|T\|K^{dn}\|WF^n\|L) + \\ &+ i\cdot\partial_H(WS^{dn}\|T^r\|s4(dn)\cdot K\|WF^n\|L) + \\ &+ i\cdot\partial_H(WS^{dn}\|T^r\|s4(ce)\cdot K\|WF^n\|L) + \\ &+ i\cdot\partial_H(WS^{dn}\|T^r\|K\|WF^n\|L)\ )\end{aligned} \tag{3}$$

(Action $c\,7(to)$ has lower priority than the other actions.)

$$= \tau\cdot[WS^{dn}\|T^r\|K\|SH^{dn}\|L] +$$
$$+ \tau\cdot[SF^{dn}\|T\|K\|WF^{n}\|L]$$

(If the message is damaged, the resulting state is the same as in the case in which the message gets lost. In both cases a time out event occurs.)

$$[WS^{dn}\|T^r\|K\|SH^{dn}\|L] = s\,2(d)\cdot[WS^{dn}\|T^r\|K\|SA^{1-n}\|L] \tag{4}$$

$$[WS^{dn}\|T^r\|K\|SA^{1-n}\|L] = c\,6(ac)\cdot[WS^{dn}\|T^r\|K\|WF^{1-n}\|L^{ac}] \tag{5}$$

$$\begin{aligned}[WS^{dn}\|T^r\|K\|WF^{1-n}\|L^{ac}] = {} & \tau\cdot[RH^{1-n}\|T^r\|K\|WF^{1-n}\|L] + \\ & + \tau\cdot[SF^{dn}\|T^r\|K\|WF^{1-n}\|L] + \\ & + \tau\cdot[SF^{dn}\|T\|K\|WF^{1-n}\|L]\end{aligned} \tag{6}$$

$$[RH^{n}\|T^r\|K\|WF^{n}\|L] = \sum_{d\in D} r\,1(d)\cdot[SF^{dn}\|T^r\|K\|WF^{n}\|L] \tag{1a}$$

$$[SF^{dn}\|T^r\|K\|WF^{n}\|L] = c\,3(dn)\cdot[WS^{dn}\|T^r\|K^{dn}\|WF^{n}\|L] \tag{2a}$$

$$[SF^{dn}\|T^r\|K\|WF^{1-n}\|L] = c\,3(dn)\cdot[WS^{dn}\|T^r\|K^{dn}\|WF^{1-n}\|L] \tag{7}$$

$$[SF^{dn}\|T\|K\|WF^{1-n}\|L] = c\,3(dn)\cdot[WS^{dn}\|T^r\|K^{dn}\|WF^{1-n}\|L] \tag{7a}$$

$$\begin{aligned}[WS^{dn}\|T^r\|K^{dn}\|WF^{1-n}\|L] = {} & \tau\cdot[SF^{dn}\|T^r\|K\|WF^{1-n}\|L] + \\ & + \tau\cdot[WS^{dn}\|T^r\|K\|SA^{1-n}\|L]\end{aligned} \tag{8}$$

Now observe that the processes of equations 1 and 1a, 2 and 2a, and 7 and 7a are identical. This means we have derived that X $(= [S\|T\|K\|R\|L])$ satisfies the system of recursion equations in Table 11.

(0)	$X = X_1^0$		
(1)	$X_1^n = \sum_{d\in D} r\,1(d)\cdot X_2^{dn}$	(5)	$X_5^{dn} = c\,6(ac)\cdot X_6^{dn}$
(2)	$X_2^{dn} = c\,3(dn)\cdot X_3^{dn}$	(6)	$X_6^{dn} = \tau\cdot X_1^{1-n} + \tau\cdot X_7^{dn}$
(3)	$X_3^{dn} = \tau\cdot X_2^{dn} + \tau\cdot X_4^{dn}$	(7)	$X_7^{dn} = c\,3(dn)\cdot X_8^{dn}$
(4)	$X_4^{dn} = s\,2(d)\cdot X_5^{dn}$	(8)	$X_8^{dn} = \tau\cdot X_5^{dn} + \tau\cdot X_7^{dn}$

TABLE 11. Recursion equations for X

This finishes the first part of the proof. In the second part we will abstract from the communications at ports 3 and 6. Because $PAR = \tau_I(X) = \tau_I(X_1^0)$, it is enough to show that

$$\tau_I(X_1^0) = \sum_{d \in D} r1(d) \cdot s2(d) \cdot \tau_I(X_1^0).$$

For d and n fixed, variables X_2^{dn} and X_3^{dn} form a conservative cluster from I. Hence we can apply CFAR:

$$\tau_I(X_2^{dn}) = \tau \cdot \tau_I(X_4^{dn}).$$

Variables X_5^{dn}, X_6^{dn}, X_7^{dn} and X_8^{dn} (d and n fixed) also form a conservative cluster from I. CFAR gives:

$$\tau_I(X_5^{dn}) = \tau \cdot \tau_I(X_1^{1-n}).$$

We use these two results in the following derivation:

$$\begin{aligned}\tau_I(X_1^n) &= \sum_{d \in D} r1(d) \cdot \tau_I(X_2^{dn}) \\ &= \sum_{d \in D} r1(d) \cdot \tau \cdot \tau_I(X_4^{dn}) \\ &= \sum_{d \in D} r1(d) \cdot s2(d) \cdot \tau_I(X_5^{dn}) \\ &= \sum_{d \in D} r1(d) \cdot s2(d) \cdot \tau_I(X_1^{1-n}).\end{aligned}$$

Substituting this equation in itself gives:

$$\tau_I(X_1^0) = \sum_{d \in D} r1(d) \cdot s2(d) \cdot \sum_{e \in D} r1(e) \cdot s2(e) \cdot \tau_I(X_1^0) \quad \text{and}$$

$$\tau_I(X_1^1) = \sum_{d \in D} r1(d) \cdot s2(d) \cdot \sum_{e \in D} r1(e) \cdot s2(e) \cdot \tau_I(X_1^1).$$

Due to the Recursive Specification Principle we have:

$$\tau_I(X_1^0) = \tau_I(X_1^1).$$

Hence

$$\tau_I(X_1^0) = \sum_{d \in D} r1(d) \cdot s2(d) \cdot \tau_I(X_1^0),$$

which is the desired result. □

REMARK. For the modelling of time outs in the PAR protocol the use of the priority operator is not essential. We sketch an alternative. If a frame gets lost in one of the channels then one can say that this event in a sense *causes* a time out. This causal relationship can be expressed in process algebra by means of a communication between the channel and the pair sender/timer. For channels K and L the specifications then become as shown in Table 12.

$$\begin{aligned}\overline{K} &= \sum_{f\in D\times B} r3(f)\cdot\overline{K}^f\\ \overline{K}^f &= (i\cdot s4(f)+i\cdot s4(ce)+i\cdot s7(to))\cdot\overline{K}\end{aligned}$$

$$\begin{aligned}\overline{L} &= r6(ac)\cdot\overline{L}^{ac}\\ \overline{L}^{ac} &= (j\cdot s5(ac)+j\cdot s5(ce)+j\cdot s7(to))\cdot\overline{L}\end{aligned}$$

TABLE 12. Specification for channels $\overline{K}$ and $\overline{L}$

In a time out event *three* processes participate: the timer, the sender and a channel. This means that when dealing with time outs we have ternary communication at port 7.

$$\gamma(s7(to),s7(to)) = ss7(to) \quad \gamma(s7(to),r7(to)) = sr7(to)$$
$$\gamma(s7(to),sr7(to)) = c7(to) \quad \gamma(r7(to),ss7(to)) = c7(to)$$

This leads to a slightly bigger set of unsuccessful communications:

$$\overline{H} = H\cup\{ss7(to),sr7(to)\}.$$

The alternative specification of the PAR protocol now becomes.

$$\boxed{\overline{PAR} = \tau_I\circ\partial_{\overline{H}}(S\|\overline{K}\|R\|\overline{L})}$$

One can prove that $PAR=\overline{PAR}$. In [11], essentially the above idea is used to specify a simple version of the PAR protocol.

5.2. *Asymmetric communication*

Consider the situation where channel K contains a frame and the receiver is doing some other things and reads the datum from K only after a long time. Now one can consider it to be unnatural that during this whole period the datum keeps 'floating' in K and does not disappear. In a more realistic approach we would assume that if a datum is contained in channel K, either this is read by process R, or it gets lost if R is not willing to receive. Formally we can model this in process algebra by not encapsulating $s4(d)$ actions, but give them a lower priority than the corresponding $c4(d)$ actions. This mechanism is called *put mechanism* in [3]. One can prove that the ABP and the PAR protocol are invariant under the use of the put mechanism.

REFERENCES

1. J.C.M. BAETEN, J.A. BERGSTRA, J.W. KLOP (1986). Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae IX(2),* 127-168.
2. K.A. BARTLETT, R.A. SCANTLEBURY, P.T. WILKINSON (1969). A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM 12,* 260-261.
3. J.A. BERGSTRA (1985). *Put and Get, Primitives for Synchronous Unreliable Message Passing,* Logic Group Preprint Series Nr. 3, CIF, State University of Utrecht.
4. J.A. BERGSTRA, J.W. KLOP (1986). Verification of an Alternating Bit Protocol by means of process algebra. W. BIBEL, K.P. JANTKE (eds.). *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31,* Akademie-Verlag, Berlin, 9-23. Also appeared as CWI Report CS-R8404, Centre for Mathematics and Computer Science, Amsterdam, 1984.
5. J.A. BERGSTRA, J.W. KLOP (1986). Process algebra: specification and verification in bisimulation semantics. M. HAZEWINKEL, J.K. LENSTRA, L.G.L.T. MEERTENS (eds.). *Mathematics and Computer Science II,* CWI Monograph 4, North-Holland, Amsterdam, 61-94.
6. R.J. VAN GLABBEEK (1987). Bounded nondeterminism and the approximation induction principle in process algebra. F.J. BRANDENBURG, G. VIDAL-NAQUET, M. WIRSING (eds.). *Proc. STACS 87,* LNCS 247, Springer-Verlag, 336-347.
7. R.J. VAN GLABBEEK, F.W. VAANDRAGER (1988). *Modular Specifications in Process Algebra - With Curious Queues,* CWI Report CS-R8821, Centre for Mathematics and Computer Science, Amsterdam. Extended abstract to appear in *Proceedings of the METEOR Workshop on Algebraic Methods: Theory, Tools and Applications,* LNCS, Springer-Verlag.
8. J.Y. HALPERN, L.D. ZUCK (1987). *A Little Knowledge Goes a Long Way: Simple Knowledge-based Derivations and Correctness Proofs for a Family of Protocols* (extended abstract), IBM Almaden Research Center.
9. C.P.J. KOYMANS, J.C. MULDER (1989). *A Modular Approach to Protocol Verification using Process Algebra.* This volume.
10. K.G. LARSEN, R. MILNER (1987). A complete protocol verification using relativized bisimulation. TH. OTTMANN (ed.). *Proceedings 14th ICALP, Karlsruhe,* LNCS 267, Springer-Verlag, 126-135.
11. J. PARROW (1985). *Fairness Properties in Process Algebra - with Applications in Communication Protocol Verification,* DoCS 85/03, Ph.D. Thesis, Department of Computer Systems, Uppsala University.
12. T. STREICHER (1987). *A Verification Method for Finite Dataflow Networks with Constraints Applied to the Verification of the Alternating Bit Protocol,* Report MIP-8706, Fakultät für Mathematik und Informatik, Universität Passau.
13. A.S. TANENBAUM (1981). *Computer Networks,* Prentice-Hall International.
14. F.W. VAANDRAGER (1986). *Verification of Two Communication Protocols by means of Process Algebra,* CWI Report CS-R8608, Centre for Mathematics and Computer Science, Amsterdam.
15. D. VERGAMINI (1986). *Verification by means of Observational Equivalence on Automata,* Report 501, INRIA, Centre Sophia-Antipolis, Valbonne Cedex.