Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

F.W. Vaandrager

Verification of two communication protocols
by means of process algebra

Computer Science/Department of Software Technology Report CS-R8608 January

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 two Communication Protocols by means of Process Algebra

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

A Positive Acknowledgement with Retransmission protocol, and a One Bit Sliding Window protocol are verified in the framework of ACP_τ, the algebra of communicating processes with silent steps, augmented with some additional axioms. We present the Cluster Fair Abstraction Rule (CFAR), which is a generalization of Koomen's Fair Abstraction Rule (KFAR), and show that CFAR can be derived from $KFAR_1$. We introduce the notion of redundancy in a context, which makes it possible to use trace theoretic arguments in process algebra calculations. For the verification of the second protocol, we use the technique of local replacement. In this technique a concurrent system is simplified by repeated replacement of components, replacements which leave the behaviour of the system invariant.

1980 Mathematical Subject Classification: 68B10, 68C01, 68D25, 68F20.
1982 CR Categories: F.1.1, F.1.2, F.3.2, F.4.3, C.2.2.
Key Words & Phrases: process algebra, concurrency, communication protocol, verification, fairness, trace set, redundancy, local replacement.
Note: This report is a revised and extended version of the thesis submitted by the author in partial fullfillment of the requirements for the Master's degree in Mathematics and Computer Science at the State University of Leiden. The thesis was written under the guidance of Prof. dr. J.A. Bergstra and Prof. dr. A. Ollongren.
This report will be submitted for publication elsewhere. Partial support received from the European Communities under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (Meteor).
Andrew S. Tanenbaum, Computer Networks, 1981, pp. 152, 154, reproduced by permission of Prentice-Hall, Englewood Cliffs, NJ.

INTRODUCTION

In this paper, we analyse communication protocols. A communication protocol is a set of rules and prescriptions, that is given in order to achieve reliable, efficient communication between two (or more) processors, connected by communication channels. In BERGSTRA & KLOP [7], a simple version of the Alternating Bit protocol was verified within the framework of process algebra. In this paper we study two communication protocols, which were described in TANENBAUM [12]: a Positive Acknowledgement with Retransmission (PAR) protocol, and a One Bit Sliding Window (OBSW) protocol. Among the existing communication protocols, PAR and OBSW are rather simple ones, and the only reason for studying them lies in the fact that they allow us to develop the theory of process algebra.

Although this is not an introductory paper about process algebra, we think that, in principle, someone who is not acquainted with process algebra can read it. In the first section we give a short review of the theory of process algebra. A more comprehensive introduction is presented in, for example, BERGSTRA & KLOP [8].

In §2, we discuss Koomen's Fair Abstraction Rule (KFAR), a proof rule which is vital in algebraic computations for system verification. KFAR says that a process 'does not get stuck' in a cycle of internal actions. KFAR is parametrized by $k \geqslant 1$, indicating the length of the internal cycle. We formulate a generalization of KFAR, the Cluster Fair Abstraction Rule (CFAR), which says that a process does not get stuck in certain graph structures, which we call 'conservative clusters'. We prove that CFAR can be derived from $KFAR_1$. As a consequence of this, axioms $KFAR_k$ are redundant for $k \geqslant 2$ (in any practical case, to be more precise). In §3 and §4 we specify and verify the PAR-protocol.

Report CS-R8608
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In these sections our main goal is to show how a verification can be accomplished within the framework of process algebra; we do not describe in a detailed way how the process algebra specification of the PAR-protocol is related to Tanenbaum's description.

However, this is done for the OBSW-protocol in §5. In this section we use the state operator Λ_σ^m to translate Tanenbaum's computer program into process algebra.

In §6 we introduce the notion of 'redundancy in a context'. With the help of this notion, which is of a trace theoretic nature, a specification can sometimes be simplified.

In §7 we verify the OBSW-protocol. For this verification we use the technique of local replacement. In this technique the complexity of a concurrent system is reduced by repeated replacement of components, replacements which leave (after abstraction) the behaviour of the system invariant.

TABLE OF CONTENTS

§1 PROCESS ALGEBRA

In this section, we give a brief review of a number of topics in the theory of process algebra. First we discus the axiom system ACP_τ, the algebra of communicating processes with silent steps (see BERGSTRA & KLOP [6]. Often we expand the system ACP_τ with a number of operators and axioms. These are reviewed in section 1.2 - 1.12, except for Koomen's Fair Abstraction Rule (KFAR), which will be discussed in more detail in section 2.

In the analysis of both protocols make use of the axiom system ACP_θ, the algebra of communicating processes with priorities, as described in BAETEN, BERGSTRA & KLOP [3]. We review this axiom system in section 1.13.

1.1 ACP_τ

1.1.1 Signature.

$\mathbb{S}$ (Sorts):	A	(a finite set of atomic actions)
	P	(the set of processes; $A \subseteq P$)
$\mathbb{F}$ (Functions):	$+ : P \times P \to P$	(alternative composition or sum)
	$\cdot : P \times P \to P$	(sequential composition or product)
	$\| : P \times P \to P$	(parallel composition or merge)
	$\lfloor\!\lfloor : P \times P \to P$	(left-merge)
	$\mid : P \times P \to P$	(communication merge; $\mid : A \times A \to A \cup \{\delta\}$ is given)
	$\partial_H : P \to P$	(encapsulation ; $H \subseteq A$)
	$\tau_I : P \to P$	(abstraction ; $I \subseteq A$)
$\mathbb{C}$ (Constants):	$\delta \in P - A$	(deadlock)
	$\tau \in P - A$	(silent or internal action)

1.1.2 Note. We will use the abbreviations $A_\delta = A \cup \{\delta\}$, $A_\tau = A \cup \{\tau\}$ and $A_{\tau,\delta} = A \cup \{\tau,\delta\}$.

1.1.3 Axioms. These are presented in table 1.
Here $a,b \in A_\delta; x,y,z \in P; H \subseteq A$ and $I \subseteq A$.

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		
$a\|b = b\|a$	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
$\partial_H(xy) = \partial_H(x)\cdot\partial_H(y)$	D4	$\tau_I(xy) = \tau_I(x)\cdot\tau_I(y)$	TI5

TABLE 1.

1.1.4 DEFINITION: The set of Basic Terms, BT, is defined inductively as follows:
(i) $\tau,\delta \in BT$
(ii) $x \in BT \Rightarrow \tau x \in BT$
(iii) $a \in A \ \& \ x \in BT \Rightarrow ax \in BT$
(iv) $x,y \in BT \Rightarrow x+y \in BT$

We call a ACP_τ-term closed if it contains no process variables. Such a term may contain atomic steps though (variables ranging over A). The set BT, together with the following theorem, allow us to use induction in proofs.

1.1.5 Elimination Theorem.
Let t be a closed term in the signature of ACP_τ. Then $\exists t' \in BT: ACP_\tau \vdash t = t'$.

PROOF: See BERGSTRA & KLOP [6].

1.2 Standard Concurrency. (SC)
Often we expand the system ACP_τ with the following axioms of Standard Concurrency (see table 2).

$(x \mathbin{\llcorner} y) \mathbin{\llcorner} z = x \mathbin{\llcorner} (y \| z)$	SC1
$(x \mid ay) \mathbin{\llcorner} z = x \mid (ay \mathbin{\llcorner} z)$	SC2
$x \mid y = y \mid x$	SC3
$x \| y = y \| x$	SC4
$x \mid (y \mid z) = (x \mid y) \mid z$	SC5
$x \| (y \| z) = (x \| y) \| z$	SC6

TABLE 2.

1.3 Handshaking Axiom. (HA)
HA says that all communications are binary

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

If we adopt HA + SC, then it is easy to prove the following Expansion Theorem. Let $x_1, \ldots, x_n$ be given processes, and let $\overrightarrow{x}^j$ be the merge of all $x_1, \ldots, x_n$ except x_i, $\overrightarrow{x}^{j,j}$ be the merge of all $x_1, \ldots, x_n$ except x_i and x_j, then the *Expansion Theorem* (ET) is

$$\text{(ET)} \quad x_1 \| x_2 \| \ldots \| x_n = \sum_{1 \leq i \leq n} x_i \mathbin{\llcorner} \overrightarrow{x}^j + \sum_{1 \leq i < j \leq n} (x_i \mid x_j) \mathbin{\llcorner} \overrightarrow{x}^{j,j}$$

in words: if you merge a number of processes, you can start with an action from one of them or with a communication between two of them.

1.4 Projection. (PR)
Reasoning about processes often uses a projection operator

$$\pi_n : P \to P \quad (n \geq 1),$$

which "cuts of" processes at depth n (after doing n steps), but with the understanding that τ-steps are "transparent", i.e. a τ-step does not raise the depth. Axioms for π_n are given in table 3.

$\pi_n(a) = a$	PR1	$\pi_n(\tau) = \tau$	PRT1
$\pi_1(ax) = a$	PR2	$\pi_n(\tau x) = \tau\pi_n(x)$	PRT2
$\pi_{n+1}(ax) = a\pi_n(x)$	PR3		
$\pi_n(x+y) = \pi_n(x) + \pi_n(y)$	PR4		

TABLE 3.

1.5 Specifications.

1.5.1 DEFINITION: A *(recursive) specification* $E = \{E_j : j \in J\}$ is a set of equations in the language of ACP_τ with variables $\{X_j : j \in J\}$, such that equation E_j has the form

$$X_j = T_j$$

where T_j is a finite ACP_τ-term (with finitely many variables), and the index set J contains a designated element j_0.

1.5.2 DEFINITION: Let J be a set, E a recursive specification indexed by J, and let $\{x_j : j \in J\}$ be processes. Put $x = x_{j_0}$, $\mathbb{X} = \{x_j : j \in J - \{j_0\}\}$.

1. *x is a solution of E with parameters $\mathbb{X}$*, notation $E(x, \mathbb{X})$, if substituting the x_j for variables X_j in E gives only true statements about processes $\{x_j : j \in J\}$.
2. *x is a solution of E*, notation $E(x, -)$, if there are processes $\mathbb{X} = \{x_j : j \in J - \{j_0\}\}$ such that $E(x, \mathbb{X})$.

1.5.3 DEFINITION: Let T be an open ACP_τ-term without an abstraction operator τ_I. An occurrence of a variable X in T is *guarded* if T has a subterm of the form aM, with $a \in A_\delta$ (so $a \neq \tau$), and this X occurs in M. Otherwise, the occurrence is *unguarded.*
Let $E = \{E_j : j \in J\}$ be a specification without an abstraction operator τ_I, and let $i, j \in J$. We define

$$X_i \xrightarrow{u} X_j \Leftrightarrow X_j \text{ occurs unguarded in } T_i,$$

and we call E *guarded* if relation $\xrightarrow{u}$ is well-founded (i.e. there is no infinite sequence $X_{j_1} \xrightarrow{u} X_{j_2} \xrightarrow{u} X_{j_3} \xrightarrow{u} \cdots$).
(For these definitions, also see BAETEN, BERGSTRA & KLOP [4]).

1.5.4 DEFINITION: Let $E = \{E_j : j \in J\}$ be a specification, and let $j \in J$. An *expansion* of X_j is an open ACP_τ-term obtained by a series of substitutions of T_i for occurrences of X_i in E_j. For a more precise definition, see BAETEN, BERGSTRA & KLOP [2], 2.7.

1.5.5 LEMMA. Let E be a guarded recursive specification in which no abstraction operator τ_I occurs, and let $j \in J$ (the index set of E). Then X_j has an expansion in which all occurrences of variables are guarded.

PROOF: Essentially, this is lemma 2.14 in BAETEN, BERGSTRA & KLOP [2]. We build up such an expansion in the following way. If in T_j, all occurrences of variables are guarded, we are done. Otherwise, substitute T_i for all unguarded X_i in T_j, and repeat this process. This must stop after finitely many steps, for otherwise we obtain by König's lemma an infinite sequence $X_j \xrightarrow{u} X_i \xrightarrow{u} \cdots$, which contradicts the well-foundedness of $\xrightarrow{u}$

1.5.6 THEOREM. Let $E = \{E_j : j \in J\}$ be a guarded recursive specification, in which no abstraction operator τ_I occurs; let $j \in J$ and let $n \geqslant 1$. Then $\pi_n(X_j)$ can be expanded to a closed finite ACP_τ-term.

PROOF: By iterated application of lemma 1.5.5.

1.6 Recursive Definition Principle. (RDP)
RDP states that each guarded specification, in which no abstraction operator τ_I occurs, has a solution.

$$\text{(RDP)}\ \frac{E \text{ guarded, no abstraction}}{\exists x\ E(x, -)}$$

1.7 Recursive Specification Principle. (RSP)
RSP says that a guarded specification, in which no τ_I occurs, has at most one solution.

$$\text{(RSP)}\ \frac{E(x, -)\quad E(y, -)}{x = y}\quad E \text{ guarded, no abstraction}$$

RDP and RSP together say that each guarded specification, with no τ_I, has a unique solution.

1.8 Approximation Induction Principle. (AIP)
AIP is a proof rule which is vital if we want to prove things about processes, which can be specified by a guarded specification with no τ_I. The rule expresses the idea that if two processes are equal to any depth, then they are equal.

$$\text{(AIP)}\ \frac{\forall n \geqslant 1\ \pi_n(x) = \pi_n(y)\quad E(x, -)}{x = y}\quad E \text{ guarded, no abstraction}$$

Notice that, as a corollary of theorem 1.5.6, AIP $\Rightarrow$ RSP

1.9 Alphabets. Define $\mathcal{A} = Pow(A)$, the set of all subsets of A. First we define the *alphabet* function for finite processes. According to the elimination theorem it is enough to give the definition for processes that can be represented by a basic term (see table 4).

1.	$\alpha(\delta) = \varnothing$
2.	$\alpha(\tau) = \varnothing$
3.	$\alpha(\tau x) = \alpha(x)$
4.	$\alpha(ax) = \{a\} \cup \alpha(x) \quad (a \in A)$
5.	$\alpha(x + y) = \alpha(x) \cup \alpha(y)$

TABLE 4.

1.9.1 Note. We have to check that

$$x = y \Rightarrow \alpha(x) = \alpha(y)$$

otherwise this definition is not correct. This is not hard to do.

1.9.2 Infinite processes. Next we define α on infinite processes:

6.	$\dfrac{E(x, -)}{\alpha(x) = \bigcup_{n=1}^{\infty} \alpha(\pi_n(x))}\ E$ guarded, no abstraction
7.	$\dfrac{E(x, -)}{\alpha(\tau_I(x)) = \alpha(x) - I}\ E$ guarded, no abstraction

1.9.3 Notes. Essential in definition 1.9.2 is the result of theorem 1.5.6, which says that $\forall n \geqslant 1$: $\pi_n(x)$ is finite. In definition 1.9.2, the partial unions $\bigcup_{n=1}^{N} \alpha(\pi_n(x))$ form an increasing sequence (with respect to the partial order $\subseteq$ on $\mathcal{A}$) as $N \to \infty$, in the finite set $\mathcal{A}$, so the sequence must be eventually constant, and the limit will always exist. More information about alphabets can be found in BAETEN, BERGSTRA & KLOP [2]. In this paper too, the following Conditional Axioms were first formulated:

1.10 Conditional Axioms. (CA)

$\dfrac{\alpha(x)\mid(\alpha(y)\cap H)\subseteq H}{\partial_H(x\parallel y)=\partial_H(x\parallel\partial_H(y))}$	CA1	$\dfrac{\alpha(x)\mid(\alpha(y)\cap I)=\varnothing}{\tau_I(x\parallel y)=\tau_I(x\parallel\tau_I(y))}$	CA2
$\dfrac{\alpha(x)\cap H=\varnothing}{\partial_H(x)=x}$	CA3	$\dfrac{\alpha(x)\cap I=\varnothing}{\tau_I(x)=x}$	CA4
$\dfrac{H=H_1\cup H_2}{\partial_H(x)=\partial_{H_1}\circ\partial_{H_2}(x)}$	CA5	$\dfrac{I=I_1\cup I_2}{\tau_I(x)=\tau_{I_1}\circ\tau_{I_2}(x)}$	CA6
$\dfrac{H\cap I=\varnothing}{\tau_I\circ\partial_H(x)=\partial_H\circ\tau_I(x)}$	CA7		

TABLE 5.

1.11 Renamings. (RN) For every function

$$f: A_{\tau,\delta}\to A_{\tau,\delta}$$

with the property that $f(\tau)=\tau$ and $f(\delta)=\delta$, we define an operator

$$\rho_f: P\to P$$

Axioms for ρ_f are given in table 6. (I is the identity)

$\rho_f(a)=f(a)\quad(a\in A_{\tau,\delta})$	RN1
$\rho_f(x+y)=\rho_f(x)+\rho_f(y)$	RN2
$\rho_f(xy)=\rho_f(x)\cdot\rho_f(y)$	RN3
$\rho_I(x)=x$	RR1
$\rho_f\circ\rho_g(x)=\rho_{f\circ g}(x)$	RR2

TABLE 6.

For $t\in A_{\tau,\delta}$, and $H\subseteq A$ we define

$$r_{t,H}: A_{\tau,\delta}\to A_{\tau,\delta}$$

to be the following function ($a\in A_\tau$):

$$r_{t,H}(a)=\begin{cases}a & \text{if } a\notin H\\ t & \text{else}\end{cases}$$

We use t_H as a notation for the operator $\rho_{r_{t,H}}$. The encapsulation operator $\partial_H=\delta_H$, and the

abstraction operator τ_I, are examples of t_H operators. The Conditional Axioms CA5, CA6 and CA7 are special cases of axiom RR2.

The following rules for t_H operators also follow immediately from axiom RR2

(1) $t_H \circ t_H(x) = t_H(x)$

(2) $t_{\{s\}} \circ s_H(x) = t_{H \cup \{s\}}(x)$

(3) $\dfrac{u \in I \ ; \ u \neq v}{t_{\{u\}} \circ v_I(x) = v_I(x)}$

The next theorem shows that the ρ_f operator is more powerful than the t_H operator.

1.11.1 THEOREM. The operator ρ_f can be written as a sequence of t_H operators $\Leftrightarrow$

$$f \in I \cup \{g \mid |g(A_{\tau,\delta})| < |A_{\tau,\delta}|\}$$

PROOF: Easy.

In this paper we will only use the t_H operator; we don't need the extra power of the ρ_f operator. The only reason to introduce the ρ_f operator was that it gives an elegant axiomization of renamings.

1.12 In BAETEN, BERGSTRA & KLOP [4], a graph model, called the standard model, is constructed for ACP_τ. All the axioms and rules we have presented thus far, hold in this model, and also Koomen's Fair Abstraction Rule which will be presented in section 2.

1.13 ACP_θ.

The axiom system ACP consists of the axioms A1-7, C1-3, CM1-9, D1-4, i.e. the left column of table 1. In ACP_θ we extend ACP with an operator θ and give some defining equations for it, to model priorities. Suppose we have a partial order $<$ on A_δ so that δ is minimal, i.e. we have for all $a,b,c \in A_\delta$

1.	$\neg(a<a)$
2.	$a<b \Rightarrow \neg(b<a)$
3.	$a<b \ \& \ b<c \Rightarrow a<c$
4.	$\delta<a$ (if $a \neq \delta$)

Let a,b,c be (atomic) actions and suppose

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

Relative to this partial order, we want to define an operator θ that models this priority

(i) $\theta(a+b) = a \ ; \ \theta(a+c) = a$;

(ii) $\theta(b+c) = b+c$

This is done in the axiom system ACP_θ (see table 7). The operator $\lhd : P \times P \to P$ is an auxilary operator which is needed in order to define θ. ACP_θ was introduced in BAETEN, BERGSTRA & KLOP [3].

1.13.1. EXAMPLES ($b<a$ and $c<a$):
(i) $\theta(a+b)=\theta(a)\triangleleft b+\theta(b)\triangleleft a=a\triangleleft b+b\triangleleft a=a+\delta=a$
(ii) $\theta(b+c)=\theta(b)\triangleleft c+\theta(c)\triangleleft b=b\triangleleft c+c\triangleleft b=b+c$
(iii) $\theta(b(a+c))=\theta(b)\cdot\theta(a+c)=b(\theta(a)\triangleleft c+\theta(c)\triangleleft a)=b(a\triangleleft c+c\triangleleft a)=b(a+\delta)=ba$

ACP_θ

$x+y=y+x$	A1	$a\triangleleft b=a$ if not $(a<b)$	P1
$x+(y+z)=(x+y)+z$	A2	$a\triangleleft b=\delta$ if $a<b$	P2
$x+x=x$	A3	$x\triangleleft yz=x\triangleleft y$	P3
$(x+y)z=xz+yz$	A4	$x\triangleleft(y+z)=(x\triangleleft y)\triangleleft z$	P4
$(xy)z=x(yz)$	A5	$xy\triangleleft z=(x\triangleleft z)y$	P5
$x+\delta=x$	A6	$(x+y)\triangleleft z=x\triangleleft z+y\triangleleft z$	P6
$\delta x=\delta$	A7		
$a\mid b=b\mid a$	C1		
$(a\mid b)\mid c=a\mid(b\mid c)$	C2		
$\delta\mid a=\delta$	C3		
$x\Vert y=x\lfloor\!\lfloor y+y\lfloor\!\lfloor x+x\mid y$	CM1	$\theta(a)=a$	TH1
$a\lfloor\!\lfloor x=ax$	CM2	$\theta(xy)=\theta(x)\cdot\theta(y)$	TH2
$ax\lfloor\!\lfloor y=a(x\Vert y)$	CM3	$\theta(x+y)=\theta(x)\triangleleft y+\theta(y)\triangleleft x$	TH3
$(x+y)\lfloor\!\lfloor z=x\lfloor\!\lfloor z+y\lfloor\!\lfloor z$	CM4		
$(ax)\mid b=(a\mid b)x$	CM5		
$a\mid(bx)=(a\mid b)x$	CM6		
$(ax)\mid(by)=(a\mid b)(x\Vert y)$	CM7		
$(x+y)\mid z=x\mid z+y\mid z$	CM8		
$x\mid(y+z)=x\mid y+x\mid z$	CM9		
$\partial_H(a)=a$ if $a\notin H$	D1		
$\partial_H(a)=\delta$ 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 7.

In BAETEN, BERGSTRA & KLOP [3] the following theorem is proved

1.13.2 THEOREM.
i) for each ACP_θ-term s there is a term t not containing $\triangleleft,\theta,\Vert,\lfloor\!\lfloor,\mid,\partial_H$ such that $\mathrm{ACP}_\theta\vdash s=t$
ii) ACP_θ is a conservative extension of ACP, i.e. for all ACP-terms s,t we have:

$$\mathrm{ACP}_\theta\vdash s=t\Rightarrow\mathrm{ACP}\vdash s=t$$

At present it is not clear whether or not ACP_θ and ACP_τ can be combined into $\mathrm{ACP}_{\tau\theta}$. However, due to theorem 1.13.2 a term like $\tau_I(s)$, with s an ACP_θ-term, makes sense: we eliminate all θ and $\triangleleft$ from s, so that it becomes an ACP-term. And when $\mathrm{ACP}_\theta\vdash s=t$, then $\mathrm{ACP}_\tau\vdash\tau_I(s)=\tau_I(t)$ for all ACP-terms s and t.

§2 Fairness

2.1 Example: A statistician performs a simple experiment: he tosses a coin until tail comes up. Let $p(tail)$ be the probability that, if he tosses the coin, tail comes up. We assume $0 < p(tail) < 1$. The behaviour of the statistician is specified by

$$S = toss \cdot (head \cdot S + tail)$$

The experiment is performed in a room. We are outside of this room and cannot observe what is going on inside, except for the fact that if tail comes up, we can hear a joyful shout of the statistician: 'tail !!'. This means that if we define

$$I = \{toss, head\}$$

the actions from I are hidden. Now the process we are interested in is specified by

$$\tau_I(S)$$

Since $0 < p(tail) < 1$, the process will perform a finite number of *head*-actions, followed by a *tail*-action. Therefore, according to our intuition

$$\tau_I(S) = \tau \cdot tail$$

What we need is an algebraic framework in which we can prove this equation.

2.2 Koomen's Fair Abstraction Rule (KFAR), introduced in Bergstra & Klop [7], expresses the idea that the nondeterministic choices made by a process are fair: a certain option is not discarded infinitely often. The following algebraic formulation is parametrized by $k \geqslant 1$, indicating the length of an internal cycle.

$$(\mathrm{KFAR}_k) \quad \frac{\forall n \in \mathbb{Z}_k \quad x_n = i_n \cdot x_{n+1} + y_n \quad (i_n \in I)}{\tau_I(x_n) = \tau \cdot \tau_I(\sum_{m \in \mathbb{Z}_k} y_m)}$$

2.3 In our example we can apply KFAR to get the desired result: because the specification of S is guarded, RDP gives that there are processes s and t such that

$$s = toss \cdot t + \delta$$

$$t = head \cdot s + tail$$

Now we can apply KFAR_2

$$\tau_I(s) = \tau \cdot \tau_I(tail + \delta) = \tau \cdot tail$$

and since s and t form an arbitrarily chosen solution

$$\tau_I(S) = \tau \cdot tail$$

2.4 Question. How to handle the case in which we know nothing about $p(tail)$ (so $p(tail)$ can be zero (poor statistician)) ? In this case the old result for $\tau_I(S)$ is certainly wrong.

Answer 1: The question shows that fairness rules are very dangerous. They yield wrong results. Therefore one should not include a fairness rule in the axiom system. KFAR is false.

Comment: Although this view can be defended, it is a bit impractical. It amounts to an admission of the proposition that process algebra (and with it any other theory of concurrency known to us) is not even able to handle in a satisfactory way the simple case of a statistician tossing a coin. Without a fairness rule protocol verification (in which the fairness of the communication channels plays an essential role) becomes impossible.

Answer 2: From a process point of view, the cases $p(tail)=0$, $0<p(tail)<1$, and $p(tail)=1$ are totally different. Therefore it is necessary to include the information we have about $p(tail)$ in the specification. If we adopt KFAR, the specification of S corresponds to the case $0<p(tail)<1$. If we want to model the situation in which we know nothing about $p(tail)$, we have to come up with another specification, for example the following.

Before the statistician starts his experiment, there are three 'possible worlds'. In the first world $p(tail)=0$, in the second one $0<p(tail)<1$, and in the third one $p(tail)=1$. As soon as the statistician starts the experiment, a choice is made between the three possible worlds (compare this with the 'collapse of the wave function' which plays a role in quantum physics). The idea yields the following specification:

$$S^* = ST + SF + SH$$

$$ST = tail$$

$$SF = head \cdot SF + tail$$

$$SH = head \cdot SH$$

Application of $KFAR_2$ gives

$$\tau_I(SF) = \tau \cdot tail$$

and $KFAR_1$ gives

$$\tau_I(SH) = \tau \cdot \delta$$

Hence

$$\begin{aligned} \tau_I(S^*) &= \tau_I(ST) + \tau_I(SF) + \tau_I(SH) = \\ &= tail + \tau \cdot tail + \tau \cdot \delta = (\text{axiom } T2) \\ &= \tau \cdot tail + \tau \cdot \delta \end{aligned}$$

This is a solution which is in accordance with our intuition.

Comment: This approach works for example 2.1, and it might work in a lot of other cases. However, it is not clear if it is always possible to model a situation in which an unfair choice occurs, by means of a fair choice. The approach of answer 2 is already problematic in the following example (due to A. Mazurkiewicz): a boy sits at the waterside and throws stones in the water. Every time before he throws he makes a choice between throwing one or two stones. We know absolutely nothing about the way this choice is made (maybe the choice is made by means of a probabilistic mechanism, but it is also possible that this is not the case). The following specification is, in presence of KFAR, certainly wrong:

$$B = one \cdot B + two \cdot B$$

With $I = \{one\}$, KFAR_1, yields

$$\tau_I(B) = \tau \cdot two \cdot \tau_I(B) = \tau \cdot two^\omega$$

This excludes the possibility that, after throwing 15 times two stones, the boy decides at every moment of choice thereafter, to throw one stone. The 'possible worlds' approach however, seems to lead unavoidably to an infinite sum

$$B^* = B1 + B2 + \cdots + B\omega$$

Infinite sums are problematic since in their presence the Approximation Induction Principle does not hold.

Answer 3: In reality certain choices are fair, other choices are unfair. A good theory must be able to deal with fair as well as unfair choices.

Comment: We can think of a lot of of possible ways to extend the theory of process algebra in such a way that we can deal with unfairness. A characteristic of all approaches is that things get more complicated. Since in the problems we will consider in this paper, the assumption that all choices are fair is reasonable, we have decided to postpone the introduction of unfairness.

2.5 EXAMPLE: The statistician of example 2.1 now throws a die (which is fair) until six comes up. The behaviour of the statistician is specified by

$$S2 = toss \cdot ((one + two + three + four + five) \cdot S2 + six)$$

The experiment is again performed in a room, and this time the only thing that can be observed by us is the joyful shout 'six !!!'. If we define

$$J = \{toss, one, two, three, four, five\}$$

the process we are interested in is specified by

$$\tau_J(S2)$$

We want to prove:

$$\tau_J(S2) = \tau \cdot six$$

However this is not so easy. The problem is that we have to do with a structure which is not a simple cycle (the normal input of KFAR).

In order to solve this problem, we will formulate the Cluster Fair Abstraction Rule (CFAR), which is a generalization of KFAR. We will prove

$$ACP_\tau + RSP + RDP + KFAR_1 + RN \vdash CFAR$$

CFAR deals with every graph structure which is a 'conservative cluster'. It will turn out that application of CFAR yields

$$\tau_J(S2) = \tau \cdot six$$

2.6 DEFINITIONS: Let $E = \{E_j : j \in J\}$ be a recursive specification, and let $I \subseteq A$. A subset C of J is called a *cluster from I in E* $\Leftrightarrow j_0 \in C$ and $\forall j \in C$:

$$\exists m \geqslant 1$$

$$\exists i_1, \ldots, i_m \in I \cup \{\tau\}$$

$$\exists f_1, \ldots, f_m \in C$$

$$\exists n \geqslant 0$$

$$\exists g_1, \ldots, g_n \in J - C \quad \text{such that } (\sum_{j \in \emptyset} x_j \equiv \delta \text{ by definition}) :$$

$$T_j = \sum_{k=1}^{m} i_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

Variables X_j with $j \in C$ are called *cluster variables.* For $i \in C$ we say

$$X_i \rightarrow X_j \Leftrightarrow X_j \text{ occurs in } T_i$$

We define

$$e(C) = \{j \in J - C \mid \exists i \in C : X_i \rightarrow X_j\}$$

Variables X_j with $j \in e(C)$ are called *exits.* $\twoheadrightarrow$ is the transitive and reflexive closure of $\rightarrow$. A cluster C from I in E is *conservative* $\Leftrightarrow$

$$\forall i \in C \, \forall j \in e(C) : X_i \twoheadrightarrow X_j$$

2.7 DEFINITION: The *Cluster Fair Abstraction Rule (CFAR)* is the following rule:

$$\text{(CFAR)} \quad \frac{E(x, \{x_j : j \in J - \{j_0\}\})}{\tau_I(x) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)} \quad \begin{array}{l} E \text{ guarded, no abstraction; } |I| \geqslant 2; \ C \text{ finite} \\ \text{conservative cluster from } I \text{ in } E \end{array}$$

2.8 THEOREM. $ACP_\tau + RDP + RSP + RN + KFAR_1 \vdash CFAR$

PROOF: Let $E = \{E_j : j \in J\}$ be a guarded specification in which no abstraction operator occurs; let $E(x, \{x_j : j \in J - \{j_0\}\})$; let $I \subseteq A$ and $\{i, i'\} \subseteq I$ $(i \neq i')$; let C be a finite conservative cluster from I in E. We have to prove

$$\tau_I(x) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

CLAIM 1. We can assume that for $j_1 \in J - C$ and $j_2 \in J$:

$$X_{j_2} \in T_{j_1} \Rightarrow j_2 \notin C$$

PROOF: Let $C' = \{j' \mid j \in C\}$ be a copy of C. We define the specification $E' = \{E'_j : j \in J \cup C'\}$ as follows:

1. for $j \in C$ $T'_j \equiv T_j$
2. for $j' \in C'$ $T'_{j'}$ can be obtained from T_j by replacing all occurrences of variables X_k $(k \in C)$ by $X_{k'}$
3. for $j \in J - C$ T'_j can be obtained from T_j by replacing all occurrences of variables X_k $(k \in C)$ by $X_{k'}$

Because E is guarded, E' is guarded too. According to RDP E' has a solution, let us say

$$E'(x', \{x'_j : j \in J \cup C' - \{j_0\}\})$$

Substitution in E' of x_j for variables X_j $(j \in J)$, and x_j for variables $X_{j'}$ $(j' \in C')$, yields true statements. Hence

$$E'(x, \{x_j : j \in J - \{j_0\}\} \cup \{x_j : j \in C\})$$

Now RSP gives us

$$x'_j = x_j \quad (j \in \{j_0\} \cup e(C))$$

This means that it is enough to prove

$$\tau_I(x') = \tau \cdot \sum_{j \in e(C)} \tau_I(x'_j)$$

because

$$\tau_I(x) = \tau_I(x') = \tau \cdot \sum_{j \in e(C)} \tau_I(x'_j) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

The observation that C is a conservative cluster from I in E', that has the desired property, finishes the proof of the claim.

CLAIM 2. In addition we can assume that

1. if, for $j \in C$,

$$T_j = \sum_{k=1}^{m} i_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l},$$

then $i_k = i$ or $i_k = \tau$ $(1 \leqslant k \leqslant m)$, and

2. $\forall j \in e(C)\ \exists h_j \in J - C$ such that $T_j = i_I(X_{h_j})$.

PROOF: Let $ce(C) = \{j'' \mid j \in e(C)\}$ be a copy of $e(C)$. Define the specification $E'' = \{E''_j : j \in J \cup ce(C)\}$ as follows

1. if, for $j \in C$,

$$T_j = \sum_{k=1}^{m} i_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l},$$

then T''_j is defined by

$$T''_j = \sum_{k=1}^{m} i''_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g''_l}$$

with for $1 \leqslant k \leqslant m$

$$i''_k = \begin{cases} i & \text{if } i_k \neq \tau \\ \tau & \text{if } i_k = \tau \end{cases}$$

2. for $j'' \in ce(C)$

$$T''_{j''} = i_I(X_j)$$

3. for $j \in J - C$

$$T''_j = T_j$$

E'' is guarded, so it has a solution

$$E''(x'', \{x''_j : j \in J \cup ce(C) - \{j_0\}\})$$

But if we substitute in $E'' : i_I(x_j)$ for variables X_j $(j \in C)$, $i_I(x_j)$ for variables $X_{j''}$ $(j'' \in ce(C))$, and x_j for variables X_j $(j \in J - C)$, we get true statements (to see this, apply RN-axioms and use claim 1). RSP gives us

$$x'' = i_I(x), \text{ and}$$

$$x''_{j''} = i_I(x_j) \; (j'' \in ce(C))$$

Now it is enough to show that

$$\tau_I(x'') = \tau \cdot \sum_{j'' \in ce(C)} \tau_I(x''_{j''})$$

because

$$\tau_I(x) = \tau_I \circ i_I(x) = \tau_I(x'') = \tau \cdot \sum_{j'' \in ce(C)} \tau_I(x''_{j''}) =$$

$$= \tau \cdot \sum_{j \in e(C)} \tau_I \circ i_I(x_j) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

C is a conservative cluster from I in E'', which has the desired properties ($ce(C)$ is the set of exits of C in E''). This finishes the proof of claim 2.

CLAIM 3. We can even assume that for $j \in C$, T_j has the form

$$T_j = \sum_{k=1}^{m} i \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

PROOF: Let $E^1 = \{E^1_j : j \in J\}$ be the following specification: for $j \in J - C : T^1_j = T_j$, and if, for $j \in C$,

$$T_j = \sum_{k=1}^{m} i_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

then

$$T^1_j = \sum_{k=1}^{m} i^1_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

with for $1 \leqslant k \leqslant m$

$$i^1_k = \begin{cases} i & \text{if } i_k \neq \tau \text{ (hence } i_k = i \text{ according to claim 2)} \\ i' & \text{if } i_k = \tau \end{cases}$$

E^1 is guarded, so there is a solution

$$E^1(x^1, \{x^1_j : j \in J - \{j_0\}\})$$

Substitution in E of $\tau_{\{i'\}}(x^1_j)$ for variables X_j $(j \in C)$, and x^1_j for variables X_j $(j \in J - C)$, yields true statements (use claims 1 + 2, and RN-axioms, especially $\tau_{\{i'\}} \circ i_I(x) = i_I(x)$).
RSP gives

$$\tau_{\{i'\}}(x^1) = x \text{ , and}$$

$$x^1_j = x_j \; (j \in e(C))$$

Hence it is enough to show

$$\tau_I(x^1) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^1)$$

because

$$\tau_I(x) = \tau_I \circ \tau_{\{i'\}}(x^1) = \tau_I(x^1) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^1) =$$

$$= \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

We define again a new specification, $E^2 = \{E_j^2 : j \in J\}$: for $j \in J - C : T_j^2 = T_j$, and if, for $j \in C$,

$$T_j = \sum_{k=1}^{m} i_k \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

then

$$T_j^2 = \sum_{k=1}^{m} i \cdot X_{f_k} + \sum_{l=1}^{n} X_{g_l}$$

E^2 is guarded, so there is a solution

$$E^2(x^2, \{x_j^2 : j \in J - \{j_0\}\})$$

Substitution in E^2 of $i_{\{i'\}}(x_j^1)$ for variables X_j $(j \in C)$, and x_j^1 for variables X_j $(j \in J - C)$ gives us also a solution (use claims 1 + 2, and the RN-axioms, especially $i_{\{i'\}} \circ i_I(x) = i_I(x)$).
According to RSP the two solutions are equal

$$i_{\{i'\}}(x^1) = x^2$$

$$x_j^1 = x_j^2 \quad (j \in e(C))$$

Now it is enough to show

$$\tau_I(x^2) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^2)$$

because

$$\tau_I(x^1) = \tau_I \circ i_{\{i'\}}(x^1) = \tau_I(x^2) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^2) =$$

$$= \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^1)$$

Observe that C is a conservative cluster from I in E^2, and that E^2 has the desired properties. This finishes the proof of claim 3.

CLAIM 4. We can even assume that for $j \in C$, T_j has the form

$$T_j = \sum_{k=1}^{m} i \cdot X_{f_k} + \sum_{j \in e(C)} X_j$$

PROOF: Suppose $\exists i \in C \, \exists j \in e(C) : X_i \not\rightarrow X_j$. Then there is an r with $1 \leqslant r < |C| \cdot |e(C)|$ such that

$$|\{(i,j) \mid i \in C, \, j \in e(C) \text{ and } X_i \rightarrow X_j \text{ in } E\}| = r$$

We will construct a specification E^e, which is exactly the same as specification E, except for the fact that one variable has got one more exit. Hence

$$|\{(i,j) \mid i \in C, \, j \in e(C) \text{ and } X_i \rightarrow X_j \text{ in } E^e\}| = r + 1$$

Specification E^e has a unique solution

$$E^e(x^e, \{x^e_j : j \in J - \{j_0\}\})$$

Claim 1, together with RSP yields

$$x^e_j = x_j \quad (j \in e(C))$$

We will show that

$$\tau_I(x^e) = \tau_I(x)$$

After that is done, it is enough to prove

$$\tau_I(x^e) = \tau \cdot \sum_{j \in e(C)} \tau_I(x^e_j)$$

since

$$\tau_I(x) = \tau_I(x^e) = \tau \cdot \sum_{j \in e(C)} \tau_I(x^e_j) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

Notice that, because we can iterate this construction, all we have to do in order to prove claim 4, is to define specification E^e, and show that

$$\tau_I(x^e) = \tau_I(x)$$

Because cluster C is conservative

$$\exists k,\ l \in C\ \exists m \in e(C) : X_k \to X_l\ ;\ X_k \not\to X_m \text{ and } X_l \to X_m$$

(suppose not, then

$$\forall k,\ l \in C\ \forall m \in e(C) : (X_k \to X_l \text{ and } X_l \to X_m) \Rightarrow (X_k \to X_m)$$

Choose $i \in C$ and $j \in e(C)$ such that $X_i \not\to X_j$ (this is possible). Because C is conservative

$$\exists n\ \exists i_1, \ldots, i_n \in C : X_i \to X_{i_1} \to \cdots \to X_{i_n} \to X_j$$

but now we can derive $X_i \to X_j$, which is a contradiction)
So we can choose k, l and m such that equations E_k and E_l are of the form

$$X_k = i \cdot X_l + T_1$$

$$X_l = T_2 + X_m$$

and term T_1 does not contain X_m.
The specifications $E^a = \{E^a_j : j \in J\}$, $E^b = \{E^b_j : j \in J\}$, $E^c = \{E^c_j : j \in J\}$, $E^d = \{E^d_j : j \in J\}$ and $E^e = \{E^e_j : j \in J\}$ are defined as follows: the equations of the specifications are the same as the equations of E, except for the equations for variable X_k. These are resp. given by

$$T^a_k = i' \cdot X_l + T_1$$

$$T^b_k = \tau \cdot X_l + T_1$$

$$T^c_k = \tau \cdot X_l + T_1 + X_m$$

$$T^d_k = i' \cdot X_l + T_1 + X_m$$

$$T^e_k = i \cdot X_l + T_1 + X_m$$

Let $E^a(x^a, -)$, $E^b(x^b, -)$, $E^c(x^c, -)$, $E^d(x^d, -)$ and $E^e(x^e, -)$. Using the same arguments as in the proof of claim 2 we can prove

$$\tau_I(x) = \tau_I \circ i_{\{i'\}}(x^a) = \tau_I(x^a) = \tau_I \circ \tau_{\{i'\}}(x^a) = \tau_I(x^b)$$

The equations E_k^b and E_l^b are

$$X_k = \tau \cdot X_l + T_1$$
$$X_l = T_2 + X_m$$

Hence

$$X_k = \tau \cdot X_l + T_1 \overset{T2}{=} \tau \cdot X_l + X_l + T_1 = \tau \cdot X_l + T_2 + X_m + T_1 =$$
$$= \tau \cdot X_l + X_l + X_m + T_1 \overset{T2}{=} \tau \cdot X_l + T_1 + X_m$$

But since E_k^c is

$$X_k = \tau \cdot X_l + T_1 + X_m$$

and all the other equations of E^b and E^c are the same, we can apply RSP and conclude

$$x^b = x^c$$

Again using the same arguments, we continue the derivation:

$$\tau_I(x^b) = \tau_I(x^c) = \tau_I \circ \tau_{\{i'\}}(x^d) = \tau_I(x^d) = \tau_I \circ i_{\{i'\}}(x^d) = \tau_I(x^e)$$

Summarizing

$$\tau_I(x) = \tau_I(x^e)$$

This finishes the proof of claim 4.

CLAIM 5. We can even assume that $|C| = 1$.

PROOF: Define specification $E^\infty = \{E_j^\infty : j \in (J - C) \cup \{j_0\}\}$ as follows: for $j \in J - C : T_j^\infty = T_j$, and

$$T_{j_0}^\infty = i \cdot X_{j_0} + \sum_{j \in e(C)} X_j$$

E^∞ is guarded, so there is a unique solution

$$E^\infty(x^\infty, \{x_j^\infty : j \in J - C\})$$

Substitution in E of x^∞ for variables X_j $(j \in C)$, and x_j^∞ for variables X_{j_0} $(j \in J - C)$ gives us a solution of E (use claim 4). Hence

$$x^\infty = x, \text{ and } x_j^\infty = x_j \quad (j \in e(C))$$

and it is enough to prove

$$\tau_I(x^\infty) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j^\infty)$$

This finishes the proof of claim 5.

Now we are able to prove theorem 2.8. Because of claim 5 we have the equation:

$$x = i \cdot x + \sum_{j \in e(C)} x_j$$

Application of KFAR_1 now gives the desired result:

$$\tau_I(x) = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)$$

This finishes the proof of theorem 2.8.

2.9 REMARK. Consider KFAR_k ($k \geqslant 1$). It will be clear that if the "exit" processes y_n ($n \in \mathbb{Z}_k$) are specifiable by guarded specifications without abstraction operator, and if $|I| \geqslant 2$, KFAR_k is a special case of CFAR.

Because in any practical case these two premisses are fullfilled, and because the rules KFAR_k ($k > 1$) were not used in the proof of CFAR, these rules can be omitted out of our axiom system. The only fairness rule needed is KFAR_1:

$$(\mathrm{KFAR}_1) \quad \frac{x = i \cdot x + y \ (i \in I)}{\tau_I(x) = \tau \cdot \tau_I(y)}$$

2.10 REMARK. One of the constraints in CFAR is that C is finite. This constraint is not essential. Let CFAR^∞ be the following rule

$$(\mathrm{CFAR}^\infty) \quad \frac{E(x, \{x_j : j \in J - \{j_0\}\})}{\tau_I = \tau \cdot \sum_{j \in e(C)} \tau_I(x_j)}$$

E guarded, no abstraction; $|I| \geqslant 2$; C conservative cluster from I in E; $e(C)$ finite

It is possible to prove

$$\mathrm{ACP}_\tau + \mathrm{RDP} + \mathrm{RSP} + \mathrm{RN} + \mathrm{KFAR}_1 + \mathbf{PR} + \mathbf{AIP} \vdash \mathrm{CFAR}^\infty$$

Because we only need CFAR in this paper, and because the proof of CFAR^∞ is more complicated than the proof of CFAR, we confined ourselves to the proof of CFAR.

2.11 REMARK: Formally CFAR can only be applied if we have to do with a conservative cluster. In practice however, most of the specifications do not contain conservative clusters. In these cases, what we mean when we say that a result is obtained by application of CFAR is that there exists a specification that is equivalent to the specification we are dealing with (because of RSP), that this specification contains a conservative cluster, and that application of CFAR on this cluster gives the result. The specification of $S2$ in example 2.5 does not contain a conservative cluster. The following system however, which is equivalent to this specification, contains a conservative cluster.

$$S2 = \mathit{toss} \cdot X + Y$$
$$X = \mathit{one} \cdot S2 + \cdots + \mathit{five} \cdot S2 + Z$$
$$Y = \delta$$
$$Z = \mathit{six}$$

Application of CFAR now gives

$$\tau_J(S2) = \tau \cdot \mathit{six}$$

§3 ARCHITECTURE OF THE PAR-PROTOCOL

In this section we describe, in terms of process algebra, a Positive Acknowledgement with Retransmission (PAR) protocol. This communication protocol is described in TANENBAUM [12]. In the protocol, in which data are transmitted only in one direction, the sender awaits a positive acknowledgement before advancing to the next data item.

3.1 The protocol can be visualised as follows:

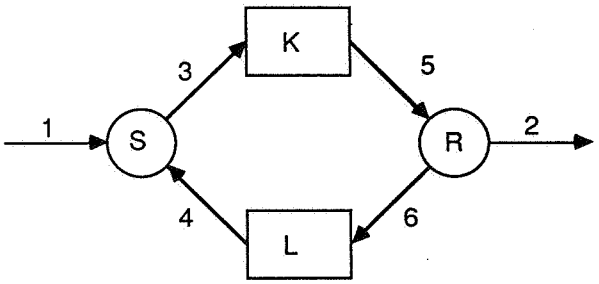


Fig. 8

There are four components:
S: Sender
K: Data transmission channel
R: Receiver
L: Acknowledgement transmission channel

The interaction of the components with their environment takes place at locations called *ports*, numbered 1 up to 6.

3.2 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. For $d \in D$ and $n \in \{0,1\}$, dn is a new datum, obtained by appending n to d. We write: $DB = \{dn \mid d \in D,\ n \in \{0,1\}\}$. Elements of DB, called frames, are communicated by channel K. The extra bit is needed for the receiver to be able to distinguish a frame that it is seeing for the first time from a retransmission.
Define $\mathbb{D} = D \cup DB \cup \{ac,ce\}$ (ac = 'acknowledgement', ce = 'checksum error'). $\mathbb{D}$ is the set of data that occur as parameter of atomic actions.

3.3 We have the following atomic actions : for $t \in \{1,2,\ldots,6\}$ there are send-, read-, and communication-actions

$st(f)$: send $f \in \mathbb{D}$ at port t

$rt(f)$: read $f \in \mathbb{D}$ at port t

$ct(f)$: communication of $f \in \mathbb{D}$ at port t

The other atomic actions are

to: time out

i, j: internal actions of channel K resp. L by which a frame gets lost.

3.4 If a message is sent into channel K or L, three things can happen:
(i) the message is communicated correctly
(ii) the message is damaged
(iii) the message is lost completely
We assume that if a message is damaged in transit, the receiver hardware will detect this when it computes the checksum (a plausible assumption we have to make).

3.5 The specification of the channels.
The channels K and L are described by the following equations. We also give the corresponding state-transition diagrams:

$$K = \sum_{f \in DB} r3(f) \cdot K^f$$

$$K^f = (s5(f) + s5(ce) + i) \cdot K \qquad f \in DB$$

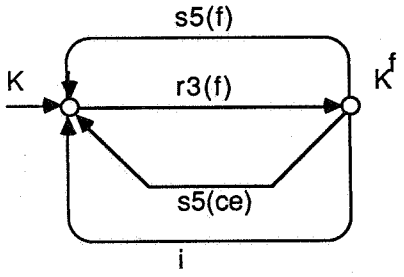


Fig. 9

$$L = r6(ac) \cdot L^{ac}$$

$$L^{ac} = (s4(ac) + s4(ce) + j) \cdot L$$

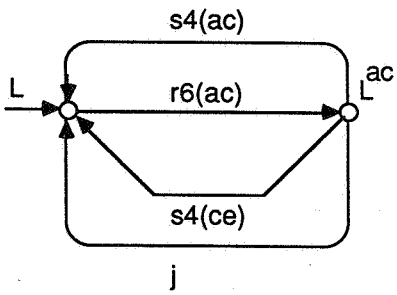


Fig. 10

3.6 REMARK. Because, as we will see later on, in the PAR-protocol it is never tried to send a message into a channel while another message is still in this channel, it is correct to model the channels as one-datum buffers.

3.7 REMARK. In BERGSTRA & KLOP [7] an "internal choice" action was used to express that a channel, after receiving input, has nondeterminate choice. With an internal choice action the specification of channel L would become

$$L = r6(ac) \cdot L^{ac}$$

$$L^{ac} = (j \cdot s4(ac) + j \cdot s4(ce) + j) \cdot L$$

In this way one avoids the unrealistic situation (for example) that the atomic action $s4(ce)$ never occurs, because the receiver never "wants" to read a damaged message. With the internal choice action one models the fact that the *channel* (and not the *receiver*) decides whether a message is communicated correctly, is damaged, or gets lost.

In the PAR-protocol however, as we will see, the receivers are never in a state in which they can read some, but not all messages. This means that we can omit the internal choice actions, and thus keep the calculation simple.

3.8 The specification of the sender S.
We use variables S, RH^n, SF^{dn}, WS^{dn} $(d \in D,\ n \in \{0,1\})$:

RH: Read a message from the Host at port 1 (the Host process, which is not specified here, furnishes the sender with data)
SF: Send a Frame at port 3
WS: Wait for Something to happen

$$S = RH^0$$

$$RH^n = \sum_{d \in D} r1(d) \cdot SF^{dn}$$

$$SF^{dn} = s3(dn) \cdot WS^{dn} \qquad d \in D,\ n \in \{0,1\}$$

$$WS^{dn} = r4(ac) \cdot RH^{1-n} + (r4(ce) + to) \cdot SF^{dn}$$

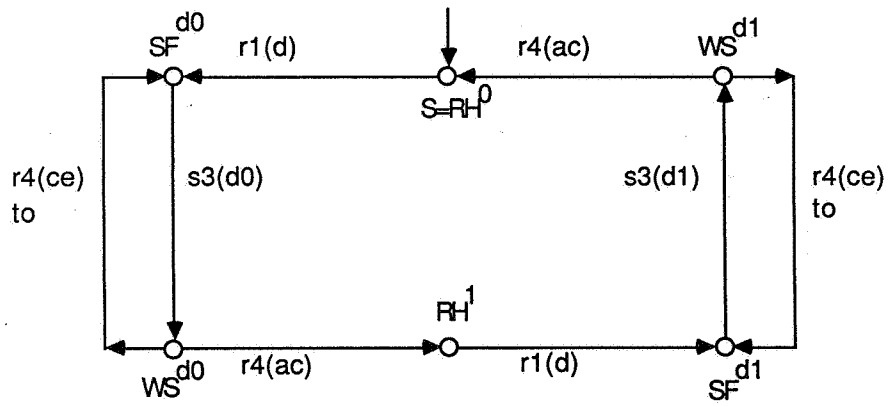


Fig. 11

After transmitting a frame, the sender waits for something to happen. There are three possibilities : an acknowledgement frame arrives undamaged, something damaged comes in, or the timer goes off. If a valid acknowledgement comes in, the sender fetches the next message, and advances the sequence number, otherwise a duplicate of the old frame is sent.

3.9 The specification of the receiver R.
We use variables R, WF^n, SA^n, SH^{dn} $(d \in D, n \in \{0,1\})$:

WF: Wait for the arrival of a Frame at port 5
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 be different from the host of the sender).

$$R = WF^0$$

$$WF^n = r5(ce) \cdot WF^n + \sum_{d \in D} r5(d, 1-n) \cdot SA^n + \sum_{d \in D} r5(d,n) \cdot SH^{dn}$$

$$SA^n = s6(ac) \cdot WF^n$$

$$SH^{dn} = s2(d) \cdot SA^{1-n} \qquad d \in D,\ n \in \{0,1\}$$

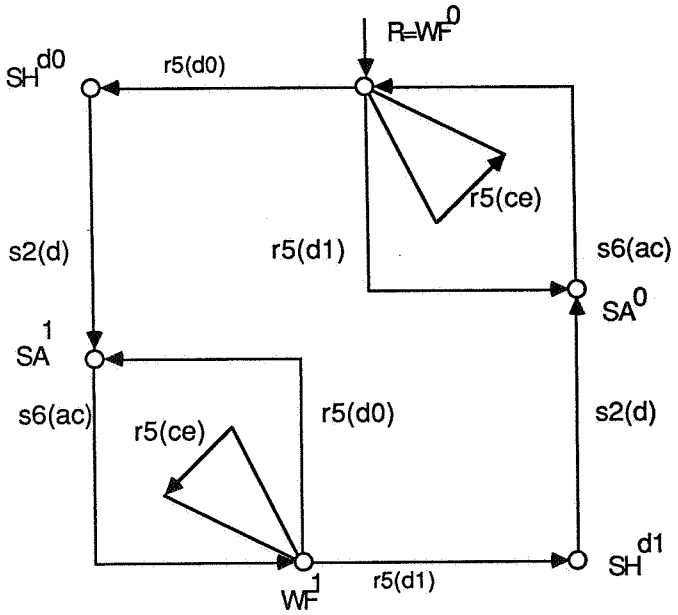


Fig. 12

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

3.10 Now we define the communication function by

$$st(f)\,|\,rt(f) = ct(f) \quad \text{for } t \in \{3,4,5,6\}, f \in \mathbb{D}$$

and all other communications give δ.
We are interested in

$$S \| K \| R \| L$$

but we want to hide unsuccesful communications.
Therefore we define

$$H = \{st(f),\ rt(f)\ |t \in \{3,4,5,6\}, f \in \mathbb{D}\}$$

and look at

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

3.11 Priority. Each time after a frame is sent, the sender S starts a timer. Because we abstract in proces algebra from the real-time behaviour of a system, we have omitted this action in our modelling. An unpleasant property of the PAR-protocol is that it requires the timeout interval to be long enough to prevent premature timeouts. 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 does arrive, the sender will mistakenly think that the just sent frame is the one being acknowledged and will not realize 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 atomic action '*to*' lower priority than every other atomic action.

So we define θ with respect to the following partial order $<$ on A_δ

(1) $\delta < a$ for $a \in A$

(2) $to < a$ for $a \in A - \{to\}$

and consider

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

3.12 Abstraction. We want to focus on the read-actions at port 1, and the send-actions at port 2. Therefore we define

$$I = \{ct(f) | t \in \{3,4,5,6\}, f \in \mathbb{D}\} \cup \{to,i,j\}$$

Now the PAR-protocol is described by

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

This is a good description because the specifications of S, K, R and L are guarded. Hence, according to RSP and RDP, these specifications have unique solutions, let's say s, k, r and l respectively. This means that there is a unique process par (par $= \tau_I \circ \theta \circ \partial_H(s \| k \| r \| l)$), which is determined by the equations for variable PAR.

§4 Verification of the PAR-protocol

Verification of the PAR-protocol 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 send the 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

4.1 Theorem. $\text{ACP}_\tau + \text{ACP}_\theta + \text{SC} + \text{HA} + \text{RDP} + \text{RSP} + \text{CA} + \text{CFAR} \vdash$

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

Proof: We define

$$I_1 = \{ct(f) | t \in \{4,5\}, f \in \mathbb{D}\} \cup \{to,i,j\}$$

$I_1 \subseteq I$, so according to axiom CA6:

$$\text{PAR} = \tau_I \circ \tau_{I_1} \circ \theta \circ \partial_H(S \| K \| R \| L)$$

In the first part of the proof we will derive a guarded system of recursion equation for

$$\tau_{I_1} \circ \theta \circ \partial_H(S \| K \| R \| L)$$

in which all terms are Basic Terms. Thereafter, in the second part, we will abstract from the other internal actions using CFAR. Throughout the following proof d ranges over D, and n ranges over $\{0,1\}$.

Fig. 13 depicts the state transition diagram that corresponds to the system of recursion equations we will derive for $\tau_{I_1} \circ \theta \circ \partial_H(S \| K \| R \| L)$.

$$\tau_{I_1} \circ \theta \circ \partial_H(S \| K \| R \| L) = \tau_{I_1} \circ \theta \circ \partial_H(RH^0 \| K \| WF^0 \| L)$$

$$X_1^n \overset{def}{\equiv} \tau_{I_1} \circ \theta \circ \partial_H(RH^n \| K \| WF^n \| L) \overset{ET}{=}$$

$$= \tau_{I_1} \circ \theta \circ \partial_H(RH^n \mathbin{\llcorner} (K \| WF^n \| L) + \cdots + L \mathbin{\llcorner} (RH^n \| K \| WF^n) +$$

$$+ (RH^n | K) \mathbin{\llcorner} (WF^n \| L) + \cdots + (WF^n | L) \mathbin{\llcorner} (RH^n \| K)) =$$

$$= \tau_{I_1} \circ \theta \circ \partial_H((\sum_{d \in D} r1(d) \cdot SF^{dn}) \mathbin{\llcorner} (K \| WF^n \| L) + \cdots +$$

$$+ (r6(ac) \cdot L^{ac}) \mathbin{\llcorner} (RH^n \| K \| WF^n) +$$

$$+ ((\sum_{d \in D} r1(d) \cdot SF^{dn}) | (\sum_{f \in DB} r3(f) \cdot K^f)) \mathbin{\llcorner} (WF^n \| L) + \cdots +$$

$$= ((r5(ce) \cdot WF^n + \cdots + \sum_{d \in D} r5(d,n) \cdot SH^{dn}) | (r6(ac) \cdot L^{ac}) \mathbin{\llcorner} (RH^n \| K)) =$$

$$= \tau_{I_1} \circ \theta \circ \partial_H(\sum_{d \in D} r1(d) \cdot (SF^{dn} \| K \| WF^n \| L) + \cdots +$$

$$+ r6(ac) \cdot (RH^n \| K \| WF^n \| L^{ac}) + \delta + \cdots + \delta =$$

$$= \tau_{I_1} \circ \theta(\sum_{d \in D} r1(d) \cdot \partial_H(SF^{dn} \| K \| WF^n \| L) + \cdots + \delta) =$$

$$= \sum_{d \in D} r1(d) \cdot \tau_{I_1} \circ \theta \circ \partial_H(SF^{dn} \| K \| WF^n \| L)$$

$$X_2^{dn} \equiv \tau_{I_1} \circ \theta \circ \partial_H(SF^{dn} \| K \| WF^n \| L) =$$

$$= c3(dn) \cdot \tau_{I_1} \circ \theta \circ \partial_H(WS^{dn} \| K^{dn} \| WF^n \| L)$$

$$X_3^{dn} \equiv \tau_{I_1} \circ \theta \circ \partial_H(WS^{dn} \| K^{dn} \| WF^n \| L) =$$

$$= \tau_{I_1} \circ \theta(to \cdot \partial_H(SF^{dn} \| K^{dn} \| WF^n \| L) +$$

$$+ i \cdot \partial_H(WS^{dn} \| K \| WF^n \| L) +$$

$$+ c5(ce) \cdot \partial_H(WS^{dn} \| K \| WF^n \| L) +$$

$$+ c5(dn) \cdot \partial_H(WS^{dn} \| K \| SH^{dn} \| L)) =$$

(*to*-action has lower priority than other actions)

$$= \tau_{I_1}(i \cdot \theta \circ \partial_H(WS^{dn} \| K \| WF^n \| L) +$$

$$+ c5(ce) \cdot \theta \circ \partial_H(WS^{dn} \| K \| WF^n \| L) +$$

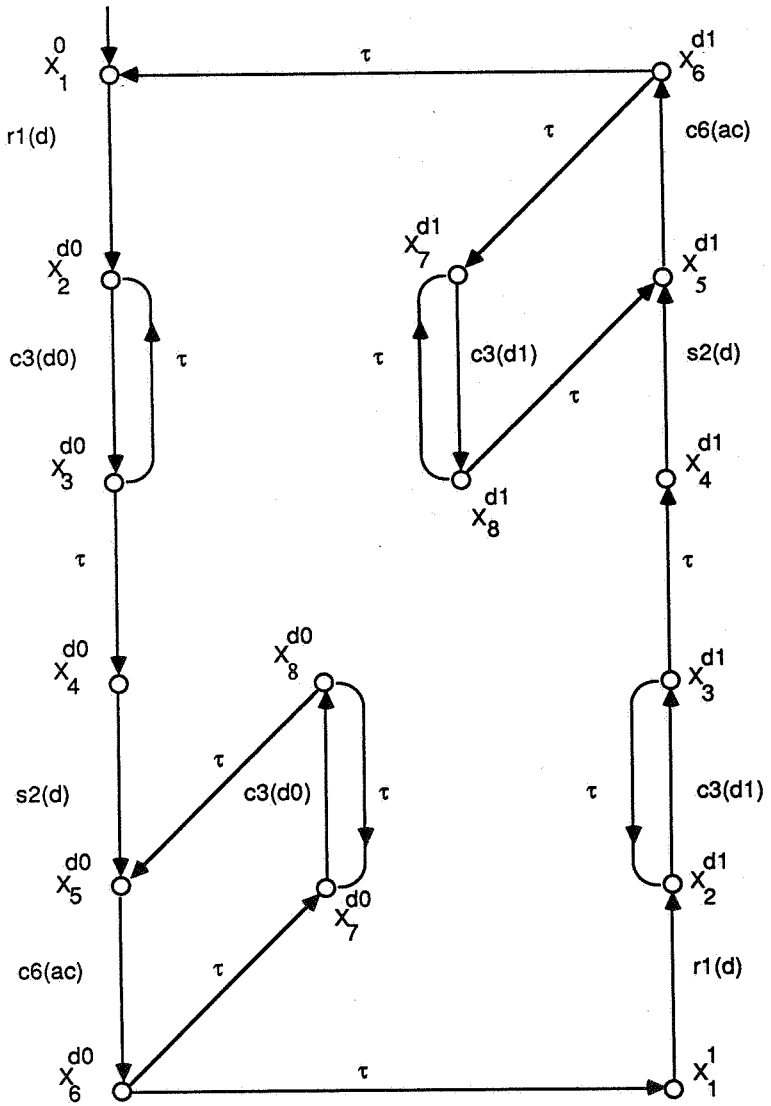

X_1^0
r1(d)
X_2^{d0}
c3(d0)
X_3^{d0}
X_4^{d0}
s2(d)
X_5^{d0}
c6(ac)
X_6^{d0}
X_7^{d0}
X_8^{d0}
X_6^{d1}
c6(ac)
X_5^{d1}
s2(d)
X_4^{d1}
X_3^{d1}
c3(d1)
X_2^{d1}
r1(d)
X_1^1
X_7^{d1}
c3(d1)
X_8^{d1}
τ

Fig. 13

$$+ c5(dn)\cdot\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L)) =$$

(if the message is damaged the resulting state is the same as in the case in which the message gets lost)

$$= \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|WF^{n}\|L) + \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L) =$$

$$= \tau\cdot\tau_{I_1}\circ\theta(to\cdot\partial_H(SF^{dn}\|K\|WF^{n}\|L) + \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L) =$$

$$= \tau\cdot\tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(SF^{dn}\|K\|WF^{n}\|L) + \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L) =$$

$$= \tau\cdot X_2^{dn} + \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L)$$

$$X_4^{dn} \equiv \tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SH^{dn}\|L) =$$

$$= s2(d)\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SA^{1-n}\|L)$$

$$X_5^{dn} \equiv \tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|SA^{1-n}\|L) =$$

$$= c6(ac)\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L^{ac})$$

$$X_6^{dn} \equiv \tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L^{ac}) =$$

$$= \tau_{I_1}(c4(ac)\cdot\theta\circ\partial_H(RH^{1-n}\|K\|WF^{1-n}\|L) +$$

$$+ c4(ce)\cdot\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L) +$$

$$+ j\cdot\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L)) =$$

$$= \tau_{I_1}(c4(ac)\cdot\theta\circ\partial_H(RH^{1-n}\|K\|WF^{1-n}\|L) +$$

$$+ c4(ce)\cdot\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L) +$$

$$+ j\cdot to\cdot\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L)) =$$

$$= \tau\cdot X_1^{1-n} + \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L)$$

$$X_7^{dn} \equiv \tau_{I_1}\circ\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L) =$$

$$= c3(dn)\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K^{dn}\|WF^{1-n}\|L)$$

$$X_8^{dn} \equiv \tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K^{dn}\|WF^{1-n}\|L) =$$

$$= \tau_{I_1}(i\cdot\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L) +$$

$$+ c5(ce)\cdot\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L) +$$

$$+ c5(dn)\cdot\theta\circ\partial_H(WS^{dn}\|K\|SA^{1-n}\|L)) =$$

$$= \tau\cdot\tau_{I_1}\circ\theta\circ\partial_H(WS^{dn}\|K\|WF^{1-n}\|L) + \tau\cdot X_5^{dn} =$$

$$= \tau\cdot\tau_{I_1}(to\cdot\theta\circ\partial_H(SF^{dn}\|K\|WF^{1-n}\|L) + \tau\cdot X_5^{dn} =$$

$$= \tau\cdot X_7^{dn} + \tau\cdot X_5^{dn}$$

Summarizing, we have found that X_1^0 $(= \tau_{1_1}\circ\theta\circ\partial_H(S\|K\|R\|L))$ satisfies the following guarded system of recursion equations:

$$X_1^n = \sum_{d \in D} r1(d) \cdot X_2^{dn}$$

$$X_2^{dn} = c3(dn) \cdot X_3^{dn}$$

$$X_3^{dn} = \tau \cdot X_2^{dn} + \tau \cdot X_4^{dn}$$

$$X_4^{dn} = s2(d) \cdot X_5^{dn}$$

$$X_5^{dn} = c6(ac) \cdot X_6^{dn}$$

$$X_6^{dn} = \tau \cdot X_1^{1-n} + \tau \cdot X_7^{dn}$$

$$X_7^{dn} = c3(dn) \cdot X_8^{dn}$$

$$X_8^{dn} = \tau \cdot X_5^{dn} = + \tau \cdot X_7^{dn}$$

TABLE 14

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

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

For fixed d and n, variables X_2^{dn} and X_3^{dn} form a guarded conservative cluster from I (in the sense of remark 2.11). 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 guarded 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:

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

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

Because of the Recursive Specification Principle

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

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.

§5 Specification of the OBSW-protocol, preliminary calculations

In this section we will give a description, in terms of process algebra, of a One Bit Sliding Window (OBSW) protocol. This protocol is described in Tanenbaum [12]. We will use the state operator Λ_σ^m to translate a computer program, which occurs in the description of the OBSW-protocol by Tanenbaum, into process algebra (In the section about the PAR-protocol we paid no attention to the relation between Tanenbaum's description of the protocol and our algebraic specification, because in that section our main goal was to show how a verification can be accomplished within the formalism of process algebra).

5.1 The essence of all *sliding window protocols* is that at any instant of time, the sender maintains a list of consecutive sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the *sending window.* Similarly, the receiver also maintains a *receiving window* corresponding to frames it is permitted to accept. The protocol we will analyse is a very simple sliding window protocol: the maximum window size is 1, and the only possible sequence numbers are 0 and 1 (One Bit).

5.2 The protocol is *full duplex.* This means that data are to be transmitted in both directions. There are two systems, A and B, both containing a sender as well as a receiver. A and B communicate by means of channels K and L (see Fig. 15). Elements of a finite data set D are to be transmitted by the protocol from port 1 to port 8, and from port 5 to port 4.

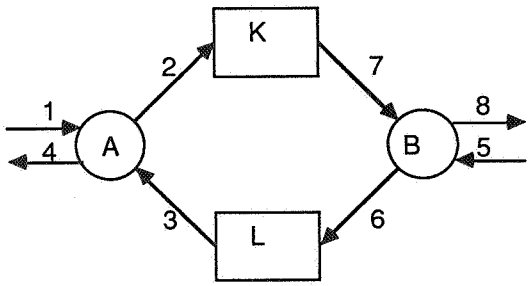


Fig. 15

5.3 The most important component of A as well as B is the *Interface Message Processor* (IMP). The IMP of system A (IMP_A) executes the program which is depicted in Fig. 16. The program for the IMP of system B (IMP_B) is slightly different, and will be presented later.

The first step we take is that we discuss the semantics of the various statements which occur in the program. Abbreviations will be introduced for the names of variables and procedures. Furthermore we reformulate a number of statements in order to reduce the number of variables.

```
const MaxSeq = 1;
type EvType = (FrameArrival, CksumErr, TimeOut);
procedure protocol4;
var NextFrameToSend: SequenceNr; {0 or 1 only}
    FrameExpected: SequenceNr;   {0 or 1 only}
    r,s: frame;                  {scratch variables}
    buffer: message;             {current message being sent}
    event: EvType;

begin
  NextFrameToSend := 0;          {initialize outbound stream}
  FrameExpected := 0;            {initialize inbound stream}
  FromHost(buffer);              {fetch message from host}
  s.info := buffer;              {prepare to send initial frame}
  s.seq := NextFrameToSend;      {frame sequence number}
  s.ack := 1 - FrameExpected;    {piggybacked ack}
  sendf(s);                      {transmit the frame}
  StartTimer(s.seq);             {start the timer running}

  repeat
    wait(event);                 {possibilities: FrameArrival, CksumErr, TimeOut}
    if event = FrameArrival then
                                 {an inbound frame made it without error}
      begin
        getf(r);                 {go get it}

        if r.seq = FrameExpected then
                                 {handle inbound frame stream}
          begin
            ToHost(r.info);      {pass the message to the host}
            inc(FrameExpected)   {invert the receiver seq number}
          end;

        if r.ack = NextFrameToSend then
                                 {handle outbound frame stream}
          begin
            FromHost(buffer);    {fetch a new message from host}
            inc(NextFrameToSend)         {invert sender seq number}
          end

      end;

    s.info := buffer;            {construct outbound frame}
    s.seq := NextFrameToSend;    {insert sequence number into it}
    s.ack := 1 - FrameExpected;  {this is seq number of last received frame}
    sendf(s);                    {transmit a frame}
    StartTimer(s.seq)            {start the timer running}
  until doomsday

end; {protocol4}
```

Fig. 16

5.3.1 The meaning of assignments, if-statements, repeat-statements and block-statement will be clear and needs no further attention. We introduce the following abbreviations:

NextFrameToSend	→	n
FrameExpected	→	e
buffer	→	b
r.seq	→	$\overline{n}$
r.ack	→	$\overline{e}$
r.info	→	$\overline{b}$

5.3.2. **procedure** FromHost (**var** m : message).
The procedure FromHost(m) fetches a message (element of D) from the host and copies it to m. Abbreviation: $Fh(m)$.

5.3.3. **procedure** ToHost (m: message).
The procedures ToHost(m) delivers the value of variable m to the host. Abbreviation: $Th(m)$.

5.3.4. **procedure** getf (**var** r : frame).
The procedure getf(r) gets an inbound frame and copies it to r. A frame is a packed record of the fields r.info, r.seq and r.ack (ranging over resp. D, B and B). Because we want to reduce the number of variables as much as possible before starting the calculations, we replace getf by a procedure

Gf(**var** m : message ; **var** p, q : SequenceNr).

This procedure gets an inbound frame, unpacks it, and copies the fields to resp. variables m, p and q.

5.3.5. **procedure** sendf (s : frame).
We replace the statements

```
s.info := buffer;
s.seq := NextFrameToSend;
s.ack := 1− FrameExpected;
sendf (s)
```

by the single statement

$Sf(b,n, 1-e)$

The procedure Sf (m : message; p, q: SequenceNr) packs the values of variables m, p and q together in a frame, and transmits this frame.

5.3.6. **procedure** StartTimer (k : SequenceNr).
This procedure starts the clock running and enables the TimeOut event. Tanenbaum does not make demands concerning the behaviour of the timers (see TANENBAUM [12], p.153):

> 'No combination of lost frames or premature timeouts can cause the protocol to deliver duplicate messages to either host, or to skip a message, or to get into a deadlock'

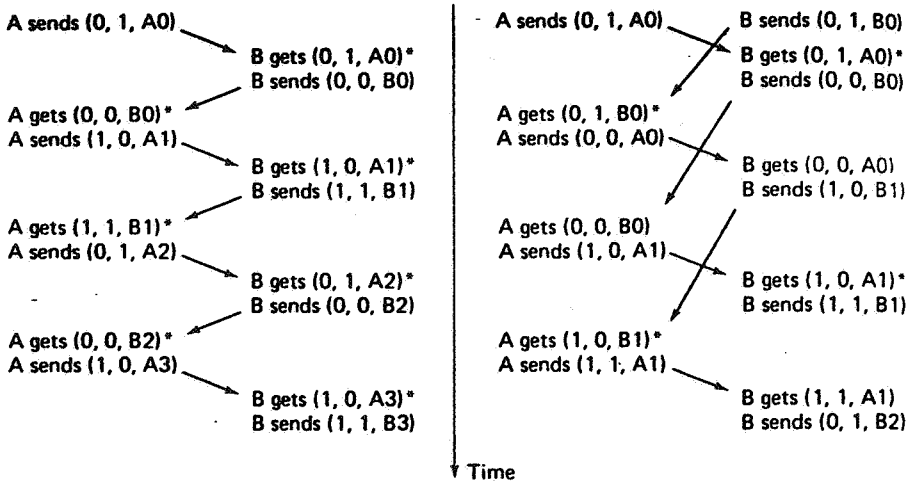
Because in process algebra we abstract from the real time behaviour of a system, the only relevant information concerning the timers is whether or not the TimeOut event is enabled. Since right before the wait(*event*) statement a StartTimer command can be found, and the wait(*event*) statement is the only place in the program where the timers can influence the process, the StartTimer command can be omitted out of the program if we assume that the TimeOut event is always enabled.

5.3.7. **procedure** wait (**var** event : EvType).
This procedure embodies the classical way to model a nondeterministic choice in a language in which nondeterministic choice is not a primitive. After the procedure call wait(*event*) the IMP sits in a tight loop waiting for something to happen. The procedure only returns when something has happened, (e.g., a frame has arrived). Upon return, the variable *event* will tell what happened; the value of *event* determines the place where the extention of the program is resumed.
We replace this construction by the statement

$$ev\,1\colon stat\,1 + ev\,2\colon stat\,2 + \cdots + evn\colon statn$$

As in process algebra '+' expresses nondeterministic choice. $ev\,1, ev\,2, \ldots, evn$ are of type EvType. In our case there are three possible values for EvType:
ce (= checksum error)
fa (= frame arrival)
to (= time out)

5.4 IMP$_B$. The program for IMP$_B$ is essentially the same as the one for IMP$_A$. However, to avoid a synchronization difficulty, the program for IMP$_B$ has to be slightly different. As pointed out by Tanenbaum, a synchronization difficulty arises if *A* and *B* both simultaneously send an initial message. Fig. 17, taken from TANENBAUM [12], illustrates the problem. In part (a), normal operation of the protocol is shown. In (b), a peculiarity is illustrated. In (a) each frame arrival brings a new message for the host. In (b) half of the frames contain duplicates, even though there are no transmission errors.



Two scenarios for protocol 4. The notation is (seq, ack, message number). An asterisk indicates where a host accepts a message.

Fig. 17

The solution Tanenbaum gives for this problem ('only one of the IMP programs should contain the 'sendf' and 'StartTimer' procedure calls outside the main loop') is incomplete. It does not cope with the situation in which the first datum sent by *A* is mangled in the communication channel, and also the timer at *A* times out too early. In this case situation (b) still can be reached. What we want is that IMP$_B$ does not undertake any action until the first message from *A* is received undamaged.

5.5 Now the flowchart of Fig. 18 represents the rewritten version of Tanenbaum's original program. The open circle corresponds to a nondeterministic choice. The intuitive meaning of the various labels is the following

PA	start of program for IMP_A
SF	send frame
WS	wait for something to happen
GF	get frame
$T1$	first test
$T2$	second test
PB	start of program for IMP_B
WF	wait for frame arrival

5.6 We translate this into process algebra as follows : all simple statements will become atomic actions, and program constructs become process algebra constructs. The recursive specification corresponding to the programs for processes A and B becomes:

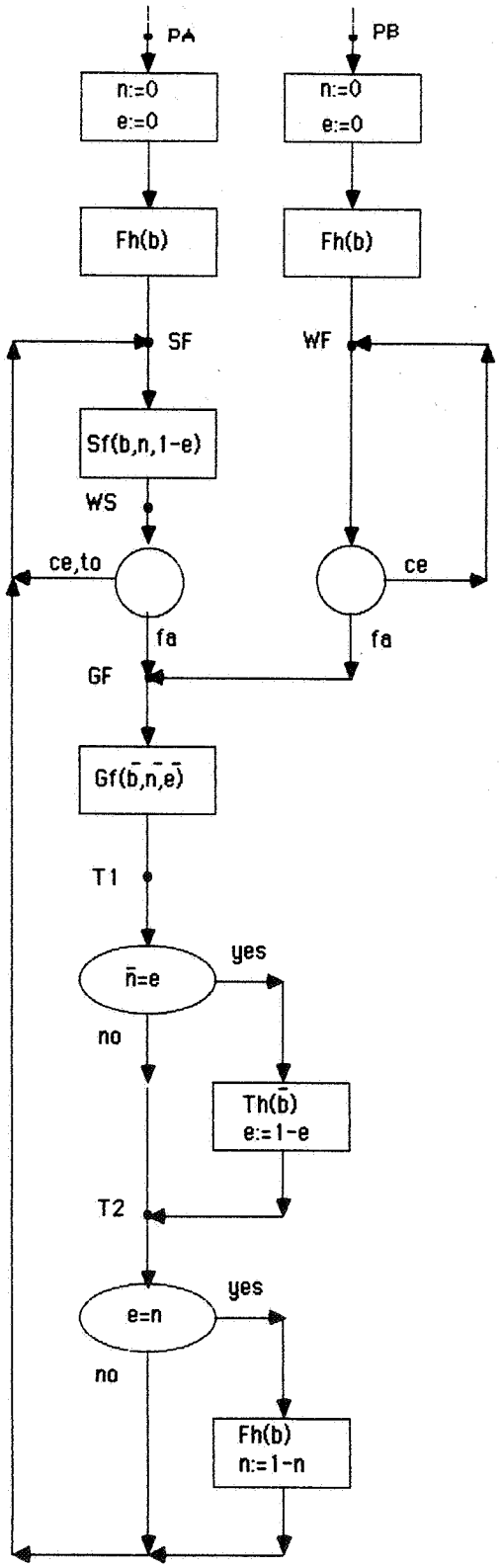

PA
PB
n:=0
e:=0
n:=0
e:=0
Fh(b)
Fh(b)
SF
WF
Sf(b,n,1-e)
WS
ce,to
ce
fa
fa
GF
Gf(b̄,n̄,ē)
T1
n̄=e
yes
no
Th(b̄)
e:=1-e
T2
e=n
yes
no
Fh(b)
n:=1-n

Fig. 18

$$PA = [n := 0]\cdot[e := 0]\cdot[Fh(b)]\cdot SF$$

$$SF = [Sf(b,n,1-e)]\cdot WS$$

$$WS = to\cdot SF + ce\cdot SF + fa\cdot GF$$

$$GF = [Gf(\bar{b},\bar{n},e)]\cdot T1$$

$$T1 = [\bar{n} = e]\cdot[Th(\bar{b})]\cdot[e := 1-e]\cdot T2 + [\bar{n}\neq e]\cdot T2$$

$$T2 = [\bar{e} = n]\cdot[Fh(b)]\cdot[n := 1-n]\cdot SF + [\bar{e}\neq n]\cdot SF$$

$$PB = [n := 0]\cdot[e := 0]\cdot[Fh(b)]\cdot WF$$

$$WF = ce\cdot WF + fa\cdot GF$$

Of course, writing down this specification does not finish the specification of the IMP's : performing an atomic action changes the value of variables, and the value of the variables determine whether or not certain actions (for example $[\bar{n} = e]$) can take place. The mechanisms used to model this is the so called state operator, which is described below.

5.7 State operator (SO). In BAETEN & BERGSTRA [1], the state operator Λ_σ^m is introduced. The formal definition of this operator looks horrifying, but the idea behind it is quite simple. First we have a certain *object* m (think of a computer). We are interested in the process which describes the behaviour of m. The object m contains a part which can be in different *states* σ (the memory of the computer). Inside the object there is also a *process* x going on (the program which is executed).
Now the principal idea is that:

(1) executing a step of the process inside object m can (from the point of view of an observer of object m) result in several possible actions. The set of possibilities depends on the state σ.
Example 1: If IMP_A performs the action $[Fh(b)]$, this results, from an external point of view, in one of the following possible actions : $\{r1(d) \mid d\in D\}$.
Example 2: If IMP_A can perform the action $[\bar{n} = e]$, but the value of $\bar{n}$ is not the same as the value of e, then this action is forbidden by the environment.

(2) executing a step of the internal process x, can, depending on the alternative chosen out of the set of possible external actions, result in a certain effect on the state σ.
Example 3: If IMP_A performs the action $[Fh(b)]$, and the chosen external action is $r1(d_0)$, then the value of variable b is changed to d_0.

Now we give the formal definition of operator Λ_σ^m :

5.8 Definition. Let M and Σ be two given finite sets, so that sets A, M, Σ are pairwise disjoint. Suppose two functions *act, eff* are given:

$$act: A\times M\times\Sigma\to Pow(A_\tau)$$

$$eff: A\times A_\tau\times M\times\Sigma\to\Sigma$$

Now we extend the signature with operators

$$\Lambda_\sigma^m : P\to P \quad (\text{for } m\in M,\sigma\in\Sigma)$$

and extend the set of axioms by ($\sum_{j\in\emptyset} x_j = \delta$) :

$$\Lambda_\sigma^m(\delta) = \delta$$

$$\Lambda_\sigma^m(\tau) = \tau$$

$$\Lambda_\sigma^m(\tau x) = \tau \cdot \Lambda_\sigma^m(x)$$

$$\Lambda_\sigma^m(ax) = \sum_{b \in act(a,m,\sigma)} b \cdot \Lambda_{eff(a,b,m,\sigma)}^m(x) \quad (a \in A)$$

$$\Lambda_\sigma^m(x+y) = \Lambda_\sigma^m(x) + \Lambda_\sigma^m(y)$$

TABLE 19

5.9 Before we define the state operators for IMP_A and IMP_B, we first take a more detailed look at the environment in which the IMP's operate. Fig. 20 is a refined version of Fig. 15.

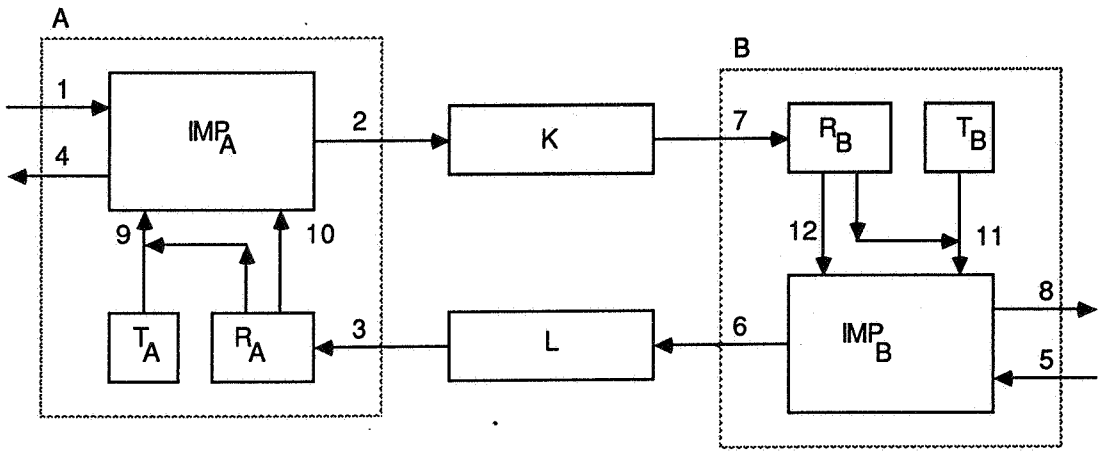


Fig. 20

In 5.3.6 we showed that we can assume that at every moment one of the timers is enabled. Because a TimeOut from the first timer cannot be distinguished from a TimeOut from the second one, the following process describes the behaviour of the *timers* in system A

$$T_A = s9(to) \cdot T_A$$

Analogously the timers of system B are described by

$$T_B = s11(to)\cdot T_B$$

The behaviour of the *receivers* R_A and R_B is specified by (with $DBB = \{dpq \mid d \in D, p \in B\}$) :

$$R_A = \sum_{f \in DBB} r3(f)\cdot R_A^f + r3(ce)\cdot R_A^{ce}$$

$$R_A^f = s9(fa)\cdot s10(f)\cdot R_A \quad (f \in DBB)$$

$$R_A^{ce} = s9(ce)\cdot R_A$$

and

$$R_B = \sum_{f \in DBB} r7(f)\cdot R_B^f + r7(ce)\cdot R_B^{ce}$$

$$R_B^f = s11(fa)\cdot s12(f)\cdot R_B \quad (f \in DBB)$$

$$R_B^{ce} = s11(ce)\cdot R_B$$

We cannot model the *communication channels* K and L as one-datum-buffers because, as we will see, in the OBSW protocol it is possible for K and L to contain more than one frame at a time. A reasonable specification seems to be a FIFO-queue with unbounded capacity. Further the behaviour of the communication channels is the same as in the PAR-protocol: if a frame is sent into a channel it can be communicated correctly, damaged, or lost completely. We give an infinite specification of the channels. It is possible to give a finite specification (see BAETEN, BERGSTRA & KLOP [4]), but that does not simplify the calculations.

$$K = K^{\epsilon} = \sum_{f \in DBB} r2(f)\cdot K^f$$

$$K^{\sigma * f} = (s7(f) + s7(ce) + i)\cdot K^{\sigma} + \sum_{g \in DBB} r2(g)\cdot K^{g * \sigma * f}$$

$$f \in DBB\,;\, \sigma \in (DBB)^*$$

$$L = L^{\epsilon} = \sum_{f \in DBB} r6(f) \cdot L^{f}$$

$$L^{\sigma * f} = (s3(f) + s3(ce) + j) \cdot L^{\sigma} + \sum_{g \in DBB} r6(g) \cdot L^{g * \sigma * f}$$

$$f \in DBB\,;\, \sigma \in (DBB)^{*}$$

5.10 State operators for IMP_A and IMP_B

There are two objects

$$M = \{A, B\}$$

Let *Var* be the set of storage elements

$$Var = \{n, e, b, \overline{n}, \overline{e}, \overline{b}\}$$

For $x \in Var$: $Dom(x)$ denotes the set of possible values of x

$$Dom(n) = Dom(e) = Dom(\overline{n}) = Dom(\overline{e}) = \{0, 1\}$$

$$Dom(b) = Dom(\overline{b}) = D$$

Let *DOM* be the set of all possible values of all variables

$$DOM = \bigcup_{x \in Var} Dom(x)$$

Now the state space Σ consists of all functions from *Var* to *DOM*, with the property that every variable has a value in its domain

$$\Sigma = \{\sigma : Var \rightarrow DOM \mid \forall x \in Var : \sigma(x) \in Dom(x)\}$$

For each $\sigma \in \Sigma$, $x \in Var$ and $\alpha \in Dom(x)$ we write $\sigma\{\alpha / x\}$ for the element of Σ which satisfies, for each $y \in Var$

$$\sigma\{\alpha / x\}(y) = \alpha, \text{ if } y = x$$

$$\sigma\{\alpha / x\}(y) = \sigma(y), \text{ if } y \neq x$$

In words, $\sigma\{\alpha / x\}$ is like σ, but for its delivering α when applied to x (this definition is adopted from DE BAKKER [5]).

The following equations define the state operator for IMP_A and IMP_B $(x \in P)$:

1. $\forall m \in M\ \forall \sigma \in \Sigma$:

$$\Lambda^{m}_{\sigma}([n := 0] \cdot x) = \tau \cdot \Lambda^{m}_{\sigma\{0 / n\}}(x)$$

$$\Lambda^{m}_{\sigma}([e := 0] \cdot x) = \tau \cdot \Lambda^{m}_{\sigma\{0 / e\}}(x)$$

$$\Lambda^{m}_{\sigma}([n := 1 - n] \cdot x) = \tau \cdot \Lambda^{m}_{\sigma\{1 - \sigma(n) / n\}}(x)$$

$$\Lambda^{m}_{\sigma}([e := 1 - e] \cdot x) = \tau \cdot \Lambda^{m}_{\sigma\{1 - \sigma(e) / e\}}(x)$$

2. $\forall\sigma\in\Sigma$:

$$\Lambda^A_\sigma([Fh(b)]\cdot x) = \sum_{d\in D} r1(d)\cdot\Lambda^A_{\sigma\{d\,/\,b\}}(x)$$

$$\Lambda^B_\sigma([Fh(b)]\cdot x) = \sum_{d\in D} r5(d)\cdot\Lambda^B_{\sigma\{d\,/\,b\}}(x)$$

3. $\forall\sigma\in\Sigma$:

$$\Lambda^A_\sigma([Th(\overline{b})]\cdot x) = s4(\sigma(\overline{b}))\cdot\Lambda^A_\sigma(x)$$

$$\Lambda^B_\sigma([Th(\overline{b})]\cdot x) = s8(\sigma(\overline{b}))\cdot\Lambda^B_\sigma(x)$$

4. $\forall\sigma\in\Sigma$:

$$\Lambda^A_\sigma([Gf(\overline{b},\overline{n},\overline{e})]\cdot x) = \sum_{d\in D}\sum_{p,q\in B} r10(d,p,q)\cdot\Lambda^A_{\sigma\{d\,/\,\overline{b}\}\{p\,/\,\overline{n}\}\{q\,/\,\overline{e}\}}(x)$$

$$\Lambda^B_\sigma([Gf(\overline{b},\overline{n},\overline{e})]\cdot x) = \sum_{d\in D}\sum_{p,q\in B} r12(d,p,q)\cdot\Lambda^B_{\sigma\{d\,/\,\overline{b}\}\{p\,/\,\overline{n}\}\{q\,/\,\overline{e}\}}(x)$$

5. $\forall\sigma\in\Sigma\ \ \forall e\in EvType$:

$$\Lambda^A_\sigma(e\cdot x) = r9(e)\cdot\Lambda^A_\sigma(x)$$

$$\Lambda^B_\sigma(e\cdot x) = r11(e)\cdot\Lambda^B_\sigma(x)$$

6. $\forall\sigma\in\Sigma\ \ \forall m\in M$:

$$\Lambda^m_\sigma([\overline{n}=e]\cdot x) = \begin{cases}\tau\cdot\Lambda^m_\sigma(x) & \text{if } \sigma(\overline{n})=\sigma(e)\\ \delta & \text{otherwise}\end{cases}$$

$$\Lambda^m_\sigma([\overline{n}\neq e]\cdot x) = \begin{cases}\tau\cdot\Lambda^m_\sigma(x) & \text{if } \sigma(\overline{n})\neq\sigma(e)\\ \delta & \text{otherwise}\end{cases}$$

$$\Lambda^m_\sigma([\overline{e}=n]\cdot x) = \begin{cases}\tau\cdot\Lambda^m_\sigma(x) & \text{if } \sigma(\overline{e})=\sigma(n)\\ \delta & \text{otherwise}\end{cases}$$

$$\Lambda^m_\sigma([\overline{e}\neq n]\cdot x) = \begin{cases}\tau\cdot\Lambda^m_\sigma(x) & \text{if } \sigma(\overline{e})\neq\sigma(n)\\ \delta & \text{otherwise}\end{cases}$$

7. $\forall\sigma\in\Sigma$:

$$\Lambda^A_\sigma([Sf(b,n,1-e)]\cdot x) = s2(\sigma(b),\sigma(n),1-\sigma(e))\cdot\Lambda^A_\sigma(x)$$

$$\Lambda^B_\sigma([Sf(b,n,1-e)]\cdot x) = s6(\sigma(b),\sigma(n),1-\sigma(e))\cdot\Lambda^B_\sigma(x)$$

This finishes the definition of the state operator. Let σ_0 be an arbitrary element of Σ. Now the process IMP_A is defined by

$$IMP_A = \Lambda^A_{\sigma_0}(PA)$$

and IMP_B is defined by

$$IMP_B = \Lambda^B_{\sigma_0}(PB)$$

5.11 LEMMA. The following specifications of IMP_A and IMP_B are equivalent to the specifications presented above ($d\in D$; $p,q\in B$):

$$IMP_A = \tau \cdot \sum_{d \in D} r1(d) \cdot SF_A^{d00}$$

$$SF_A^{dpq} = s2(d,p,1-q) \cdot WS_A^{dpq}$$

$$WS_A^{dpq} = (r9(to) + r9(ce)) \cdot SF_A^{dpq} + r9(fa) \cdot GF_A^{dpq}$$

$$GF_A^{dpq} = \sum_{e \in D} r10(e,1-q,1-p) \cdot SF_A^{dpq} +$$
$$+ \sum_{e \in D} r10(e,q,1-p) \cdot s4(e) \cdot SF_A^{dp(1-q)} +$$
$$+ \sum_{e \in D} r10(e,1-q,p) \cdot \sum_{f \in D} r1(f) \cdot SF_A^{f(1-p)q} +$$
$$+ \sum_{e \in D} r10(e,q,p) \cdot s4(e) \cdot \sum_{f \in D} r1(f) \cdot SF_A^{f(1-p)(1-q)}$$

$$IMP_B = \tau \cdot \sum_{d \in D} r5(d) \cdot WF_B^{d00}$$

$$WF_B^{d00} = r11(ce) \cdot WF_B^{d00} + r11(fa) \cdot GF_B^{d00}$$

$$SF_B^{dpq} = s6(d,p,1-q) \cdot WS_B^{dpq}$$

$$WS_B^{dpq} = (r11(to) + r11(ce)) \cdot SF_B^{dpq} + r11(fa) \cdot GF_B^{dpq}$$

$$GF_B^{dpq} = \sum_{e \in D} r12(e,1-q,1-p) \cdot SF_B^{dpq} +$$
$$+ \sum_{e \in D} r12(e,q,1-p) \cdot s8(e) \cdot SF_B^{dp(1-q)} +$$
$$+ \sum_{e \in D} r12(e,1-q,p) \cdot \sum_{f \in D} r5(f) \cdot SF_B^{f(1-p)q} +$$
$$+ \sum_{e \in D} r12(e,q,p) \cdot s8(e) \cdot \sum_{f \in D} r5(f) \cdot SF_B^{f(1-p)(1-q)}$$

PROOF: Straightforward. Use axioms ACP_τ + RDP + RSP + SO.

5.12 We define the communication function by

$$st(f) \mid rt(f) = ct(f) \text{ for } t \in \{2,3,6,7,9,10,11,12\},\ f \in DBB \cup \{ce,\ to,\ fa\}$$

and all other communications give δ.

5.13 Priority. Suppose R_A has received a frame from channel L. Now R_A gives a signal to IMP_A that a frame has arrived. IMP_A however is very busy doing other things, and doesn't notice the signal. And because the IMP doesn't fetch the frame from the receiver before the signal is noticed, and because the receiver does not read a frame from the channel before the old message is passed to the IMP, the receiver R_A does not read a new frame from the channel as long as IMP_A is busy doing other things. The frames in the channel however, are not as patient as the receiver: if no one wants to read them they just dissappear (of course the same holds the frames in channel K).

This message passing mechanism is called 'Put and Get'. In BERGSTRA [6a] it is shown that this mechanism can be modelled easily by means of the priority operator: at ports 3 and 7 send- as well as communication-actions can take place; however, if a communication-action is possible, it will occur (communication-actions have a higher priority). So we define a θ operator with respect to the following partial order $<$ on A_δ

(1) $\delta < a$ for $a \in A$
(2) $s3(f) < c3(f)$ for $f \in DBB \cup \{ce\}$
(3) $s7(f) < c7(f)$ for $f \in DBB \cup \{ce\}$

We define

$$H1 = \{r3(f) \mid f \in DBB \cup \{ce\}\}, \text{ and}$$

$$H2 = \{r7(f) \mid f \in DBB \cup \{ce\}\}.$$

Now the system consisting of components R_A and L is described by

$$\theta \circ \partial_{H1}(L \| R_A)$$

and the system R_B, K by

$$\theta \circ \partial_{H2}(K \| R_B)$$

5.14 Specification OBSW-protocol.
We define

$$H = \{st(f), rt(f) \mid t \in \{2,6,9,10,11,12\},\ f \in DBB \cup \{ce,to,fa\}\}$$

and

$$I = \{ct(f) \mid t \in \{2,3,6,7,9,10,11,12\}, f \in DBB \cup \{ce,to,fa\}\} \cup$$
$$\cup \{st(f) \mid t \in \{3,7\}, f \in DBB \cup \{ce\}\} \cup \{i,j\}$$

Now the One Bit Sliding Window protocol is described by:

$$OBSW = \tau_I \circ \partial_H(IMP_A \| T_A \| \theta \circ \partial_{H2}(K \| R_B) \| IMP_B \| T_B \| \theta \circ \partial_{H1}(L \| R_A))$$

5.15 LEMMA. Let

$$I1 = \{s3(f) \mid f \in DBB \cup \{ce\}\} \cup \{j\}, \text{ and}$$

$$H3 = \{r3(f), s3(f) \mid f \in DBB \cup \{ce\}\}$$

Let L_1 be defined as follows

$$L_1 = L_1^{\varepsilon} = \sum_{f \in DBB} r6(f) \cdot L_1^f$$

$$L_1^{\sigma * f} = (s3(f) + s3(ce) + \tau) \cdot L_1^{\sigma} + \sum_{g \in DBB} r6(g) \cdot L_1^{g * \sigma * f}$$

$$f \in DBB\,;\, \sigma \in (DBB)^*$$

Then

$$\tau_{I1} \circ \theta \circ \partial_{H1}(L \| R_A) = \partial_{H3}(L_1 \| R_A)$$

PROOF: Straightforward. In words : After abstraction it is not possible to see if a frame was lost because of a j-action by the channel, or because the receiver could not read it.

5.16 COROLLARY. Let

$$I2 = \{ct(f) \mid t \in \{2,3,6,7,9,10,11,12\}, f \in DBB \cup \{ce,to,fa\}\}$$

$$H4 = \{st(f), rt(f) \mid t \in \{2,3,6,7,9,10,11,12\}, f \in DBB \cup \{ce,to,fa\}\}$$

Let K_1 be defined by

$$K_1 = K_1^{\varepsilon} = \sum_{f \in DBB} r2(f) \cdot K^{f_1}$$

$$K_1^{\sigma * f} = (s7(f) + s7(ce) + \tau) \cdot K_1^{\sigma} + \sum_{g \in DBB} r2(g) \cdot K_1^{g * \sigma * f}$$

$$f \in DBB\,;\, \sigma \in (DBB)^*$$

Then

$$OBSW = \tau_{I2} \circ \partial_{H4}(IMP_A \| T_A \| K_1 \| R_B \| IMP_B \| T_B \| L_1 \| R_A)$$

PROOF: For reasons of symmetry an analogon of lemma 5.15 holds for K and R_B. Now apply the conditional axioms from section 1.10.

§6 REDUNDANCY IN A CONTEXT

As soon as the specification of a concurrent system is written down, one faces the question how to verify statements about this system. And if the number of components exceeds one this is not a trivial issue (as the number of publications about the subject shows).

A lot of insight into the behaviour of a system can be obtained by looking at its trace set. A trace set is the set which consists of all the sequences of actions that can be performed by a system. In this section we will give the definition of trace sets. Furthermore we will show how observations of the form 'process p never performs an action a in context $\partial_H(p \| q)$'(which are essentially of a trace-theoretic nature), can be used to simplify a specification by removing redundancies. In particular we

will show that the specification of the OBSW-protocol is redundant and can be simplified. (for more information about trace sets, see BAETEN & BERGSTRA [1], and REM [11]).

6.1 EXAMPLE: Consider the case of a theoretical computer scientist, who first drinks a cup of coffee or a cup of tea, then thinks for a while, next drinks again a cup of coffee or a cup of tea, etc. The behaviour of this scientist can be specified as follows (= coffee, = tea, = to think)

$$TCS = (\,+\,)\cdot\cdot TCS$$

There is also an automaton

$$A = \cdot A$$

Now we define the communication function by

$$\mid = \text{and} \quad \mid =$$

and all other communications give δ. We are interested in the merge of processes TCS and A, shielded off from the outside by encapsulation. Therefore we define

$$H = \{\,,\,,\,,\,\}$$

and look at

$$\partial_H(TCS \| A)$$

A simple calculation gives us that

$$\partial_H(TCS \| A) = \cdot \cdot \partial_H(TCS \| A)$$

So in a certain sense, the specification of TCS is redundant in the given context: it is useless to keep open the option of drinking tea, if there is no tea. So the following specification would do as well:

$$TCS' = \cdot \cdot TCS'$$

6.2 REMARK. A specification can also be intrinsically redundant, i.e. if you look at the specification itself (you know nothing about the context), you can come to the conclusion that some parts of it can be omitted. The following specification for example is redundant in this sense:

$$A' = \cdot A' + \cdot A'$$

the second summand can be omitted, but the solution remains the same.

However, we are not interested in this kind of redundancy here. What we want to study in this section are redundancies *in a context:* situations in which a process *cannot* perform a certain action when it is placed in a specific context. In this view the specification A′ is not redundant in context $\partial_H(TCS \| A')$. The specification $X = \partial_H(TCS \| A')$ however is redundant: first of all because TCS is redundant in context X, and in the second place because A′ is redundant.

6.3 Trace sets. Before we formally introduce the notion of trace set, we first give some definitions.

6.3.1 DEFINITION: An *alphabet* is a finite set of symbols. A *word* over an alphabet Σ is a finite list of elements from Σ. We use λ as a notation for the empty word. If σ_1 and σ_2 are words over an alphabet Σ, $\sigma_1 * \sigma_2$ (also denoted by $\sigma_1\sigma_2$) denotes the concatenation of σ_1 and σ_2. If σ is a word over an alphabet Σ, and B is a set of words over the same alphabet, we define $\sigma * B = \{\sigma * b | b \in B\}$ and $B * \sigma = \{b * \sigma | b \in B\}$. $|\sigma|$ denotes the length of σ.

6.3.2 DEFINITION: Now, we will define the *trace set* of processes which can be specified by a guarded

specification with no τ_I. A trace set is a set of words from A, so we will have a partial function

$$tr : P \to Pow(A^*)$$

On BT, we define tr inductively (see table 21).

1.	$tr(\delta) = \{\lambda\}$
2.	$tr(\tau) = \{\lambda\}$
3.	$tr(\tau x) = tr(x)$
4.	$tr(ax) = \{\lambda\} \cup a^* tr(x) \quad (a \in A)$
5.	$tr(x + y) = tr(x) \cup tr(y)$

TABLE 21.

and we extend this definition to processes which can be specified by a guarded specification with no τ_I by:

6.	$\dfrac{E(x, -)}{tr(x) = \bigcup_{n=1}^{\infty} tr(\pi_n(x))}$	E guarded, no abstraction

6.3.3 Note. Definition 6.3.2.6 is correct because trace sets are *prefix closed,* i.e. if $\sigma^*\rho$ is in some trace set ($\sigma, \rho \in A^*$), then σ is too.

6.3.4 DEFINITIONS: The following functions from $A^* \to A^*$, are defined inductively.

6.3.4.1 For $I \subseteq A$
1. $\lambda_I(\lambda) = \lambda$
2. $\lambda_I(a^*\sigma) = \lambda_I(\sigma)$ if $a \in I$
3. $\lambda_I(a^*\sigma) = a^*\lambda_I(\sigma)$ if $a \notin I$

6.3.4.2 For $t \in A$ and $H \subseteq A$
1. $t_H(\lambda) = \lambda$
2. $t_H(a^*\sigma) = t^* t_H(\sigma)$ if $a \in H$
3. $t_H(a^*\sigma) = a^* t_H(\sigma)$ if $a \notin H$

6.3.4.3 These definitions are extended to functions from $Pow(A^*) \to Pow(A^*)$ as follows. Let $B \in Pow(A^*)$. Then we define
1. $\lambda_I(B) = \{\lambda_I(\sigma) | \sigma \in B\} \quad (I \subseteq A)$
2. $t_H(B) = \{t_H(\sigma) | \sigma \in B\} \quad (t \in A, H \subseteq A)$

6.3.4.4 Now we can define the interrelation between the operators λ_I and τ_I.

7.	$\dfrac{E(x,-)}{tr(\tau_I(x)) = \lambda_I(tr(x))}$	E guarded, no abstraction

6.3.4.5 For $n \geqslant 1$ we define the function

$$\pi_n : Pow(A^*) \to Pow(A^*)$$

by

$$\pi_n(B) = \{\sigma | \quad |\sigma| \leqslant n\}$$

The following theorem follows directly from the definitions and theorem 1.5.6.

6.3.5 THEOREM. Let $x \in P$ be specifiable by a guarded specifications in which no τ_I occurs. Then

$$\forall n \geqslant 1: \quad \pi_n(tr(x)) = tr(\pi_n(x))$$

6.4 In this section we are interested in the behaviour of a process $p \in P$, when placed in a context. The types of contexts we will consider, are of the form $\partial_H(p)$ or $\partial_H(p \| q)$ $(q \in P,\ H \subseteq A)$.
We use $\partial_H(p \| \epsilon)$ as a notation for $\partial_H(p)$. So we will speak about a context $\partial_H(p \| q)$ $(q \in P \cup \{\epsilon\},\ H \subseteq A)$. In order to make the analysis easier, we demand that p is *observable* in context $\partial_H(p \| q)$: it must be possible to tell whether a certain action of $\partial_H(p \| q)$ is originated by p, and if so, by which action of p.

If, for example, we place the theoretical computer scientist of section 6.1 in a context with a number of other people and a lot of automata, and we analyze this system in order to answer the question: 'Is there a possible action sequence in which our scientist drinks a cup of tea ?', observations like 'someone is drinking something ' are very non-informative. Is our scientist drinking tea ? Or coffee ? Or is someone else drinking something ? In order to avoid these difficulties we give the following definitions.

6.4.1 DEFINITION. Let $B_1, B_2 \subseteq A$. $B_1 | B_2 = \{\gamma(b_1, b_2) \mid b_1 \in B_1; b_2 \in B_2\} - \{\delta\}$

6.4.2 DEFINITION. Let $p \in P, q \in P \cup \{\epsilon\}$ and $H \subseteq A$. We will use the notation $\alpha(\epsilon) = \varnothing$. p is *observable* in context $\partial_H(p \| q)$ if:
1. $\alpha(p) - H$, $\alpha(q) - H$ and $(\alpha(p) | \alpha(q)) - H$ are mutually disjoint.
2. $\forall a_1, a_2 \in \alpha(p)\ \forall b_1, b_2 \in \alpha(q)\ [\ \gamma(a_1, b_1) = \gamma(a_2, b_2) \in (\alpha(p) | \alpha(q)) - H\ ] \Rightarrow a_1 = a_2$

6.4.3 *Note.* Without the Handshaking Axiom the condition $(\alpha(p) - H) \cap (\alpha(p) | \alpha(q) - H) = \varnothing$ would be against our intuition. Consider, for example, the case $p = a$, $q = b$, with $a \neq b$ and $a | b = a$. Our intuition says p is observable in context $\partial_H(p \| q)$, but according to the definition it isn't.

If the second requirement is satisfied, then γ has an 'inverse' on $(\alpha(p) | \alpha(q)) - H$, i.e. if $c \in (\alpha(p) | \alpha(q)) - H$, then there is exactly one $a \in \alpha(p)$ so that a $b \in \alpha(q)$ exists with $\gamma(a, b) = c$. In this case, we put $a = \gamma^{-1}(c)$.

6.5 DEFINITIONS: Let $p \in P, q \in P \cup \{\epsilon\}, H \subseteq A$, with p observable in context $\partial_H(p \| q)$. Suppose $(\alpha(p) | \alpha(q)) - H = \{c_1, \ldots, c_n\}$. We define the trace set $v_p^{q,H}$ ($tr(\partial_H(p \| q)$ *localized to* p) by

$$\nu_p^{q,H} = \gamma^{-1}(c_1)_{\{c_1\}} \circ \cdots \circ \gamma^{-1}(c_n)_{\{c_n\}} \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q)))$$

What we do in fact is that we rename actions in the trace set of process $\partial_H(p \| q)$:
(i) Elements of $\alpha(p)$. These are the actions of p we are focusing on. So we do not rename them.
(ii) Elements of $\alpha(q)$. We are not interested in actions from q. Therefore we omit them out of the traces by means of renaming into λ.
(iii) Elements of $\alpha(p)|\alpha(q)$ These actions are renamed into their inverses, which are actions from p.
The idea of localization was, in a somewhat different form, introduced in BAETEN & BERGSTRA [1]. It is a very useful concept, which allows us to 'view' a process while it is interacting with an environment.

6.6 THEOREM. Let $p \in P$ be specifiable by a guarded specification without τ_I. Let q be ϵ, or specifiable by a guarded specification without τ_I. Let $H \subseteq A$ and let p be observable in context $\partial_H(p \| q)$. Then

$$\nu_p^{q,H} \subseteq tr(p)$$

PROOF. In case p is finite and q is finite or ϵ, the proof is easy. Using simultaneous induction on the structure of the terms (elements of $BT \cup \{\epsilon\}$) representing p and q, we can prove theorem 6.6 in a straightforward way. This part of the proof is left to the reader.

Now we use theorem 1.5.6 to reduce the general case of our theorem to the finite case. It is sufficient to prove

$$\forall n \geqslant 1 \quad \pi_n(\nu_p^{q,H}) \subseteq \pi_n(tr(p))$$

Choose an arbitrary n. Suppose $(\alpha(p)|\alpha(q)) - H = \{c_1, \ldots, c_k\}$.
We have to prove

$$\pi_n \circ \gamma^{-1}(c_1)_{\{c_1\}} \circ \cdots \circ \gamma^{-1}(c_k)_{\{c_k\}} \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q))) \subseteq \pi_n((tr(p))$$

For $t \in A$ and $H \subseteq A$, the π_m operator ($m \geqslant 1$) and the t_H operator are clearly commutative. So this is equivalent to

$$\gamma^{-1}(c_1)_{\{c_1\}} \circ \cdots \circ \gamma^{-1}(c_k)_{\{c_k\}} \circ \pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q))) \subseteq \pi_n(tr(p))$$

We cannot simply change the order of the operators π_n and $\lambda_{\alpha(q)-H}$, because in general the $\lambda_{\alpha(q)-H}$ operator doesn't preserve the length of the elements of its argument. But we claim that there exists an $m \geqslant n$ such that

$$\pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q))) = \pi_n \circ \lambda_{\alpha(q)-H} \circ \pi_m(tr(\partial_H(p \| q)))$$

First observe that, because the alphabet A is finite, the set $\pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q)))$ is finite. For each $\sigma \in \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q)))$ there is a $\sigma' \in tr(\partial_H(p \| q))$ such that $\lambda_{\alpha(q)-H}(\sigma') = \sigma$ (an 'inverse' of σ). Now choose for each element of $\pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q)))$ an inverse, and let M be the set of these inverses. So

$$\lambda_{\alpha(q)-H}(M) = \pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q)))$$

Because M is finite there exists an $m \geqslant n$ such that

$$\pi_m(M) = M$$

this means

$$\pi_n \circ \lambda_{\alpha(q)-H}(tr(\partial_H(p \| q))) = \pi_n \circ \lambda_{\alpha(q)-H} \circ \pi_m(tr(\partial_H(p \| q)))$$

Choose an $m \geqslant n$ with this property. It is enough to show

$$\gamma^{-1}(c_1)_{\{c_1\}}\circ\cdots\circ\gamma^{-1}(c_k)_{\{c_k\}}\circ\pi_n\circ\lambda_{\alpha(q)-H}\circ\pi_m(tr(\partial_H(p\|q)))\subseteq\pi_n(tr(p))$$

According to theorem 6.3.5 this is equivalent to

$$\gamma^{-1}(c_1)_{\{c_1\}}\circ\cdots\circ\gamma^{-1}(c_k)_{\{c_k\}}\circ\pi_n\circ\lambda_{\alpha(q)-H}(tr(\pi_m\circ\partial_H(p\|q)))\subseteq\pi_n(tr(p))$$

A theorem proved by VAN GLABBEEK [9] yields that the LHS can be rewritten as

$$\gamma^{-1}(c_1)_{\{c_1\}}\circ\cdots\circ\gamma^{-1}(c_k)_{\{c_k\}}\circ\pi_n\circ\lambda_{\alpha(q)-H}(tr(\pi_m\circ\partial_H(\pi_m(p)\|\pi_m(q))))\subseteq$$

(use theorem 6.3.5 and the fact that $\pi_n(B)\subseteq B$)

$$\subseteq\gamma^{-1}(c_1)_{\{c_1\}}\circ\cdots\circ(c_k)_{\{c_k\}}\circ\pi_n\circ\lambda_{\alpha(q)-H}(tr(\partial_H(\pi_m(p)\|\pi_m(q))))=$$

$$=\pi_n\circ\gamma^{-1}(c_1)_{\{c_1\}}\circ\cdots\circ\gamma^{-1}(c_k)_{\{c_k\}}\circ\lambda_{\alpha(q)-H}(tr(\partial_H(\pi_m(p)\|\pi_m(q))))=$$

$$=\pi_n(\nu^{\pi_m(q),H}_{\pi_m(p)})\subseteq\ \text{(finite case!)}$$

$$\subseteq\pi_n(tr(\pi_m(p)))=\ \text{(theorem 6.3.5)}$$

$$=\pi_n\circ\pi_m(tr(p)=\quad(n\leqslant m)$$

$$=\pi_n(tr(p))=RHS$$

because n was chosen arbitrarily, this finishes the proof of theorem 6.6.

In order to keep things simple, we will only give the definition of redundancy for a special class of specifications:

6.7 DEFINITIONS: A specification $E=\{E_j:j\in J\}$ is called *strictly linear,* if $\forall j\in J$:

$T_j=\tau$ or

$T_j=\delta$ or

$\exists m\geqslant 1$

$\exists a_1,\ldots,a_m\in A_\tau$

$\exists j_1,\ldots,j_m\in J$ such that

$$T_j=\sum_{k=1}^{m}a_k\cdot X_{j_k}$$

In the last case we say $a_k\cdot X_{j_k}$ is a *summand* of T_j, and also of E. Furthermore we say $X_{j_k}\in T_j$. It is perfectly legal to think of strictly linear specifications as graphs.

The equations $X_j=\tau$ are introduced to keep notation and proofs simple. In practical cases the variable X_j and equations E_j for which $T_j=\tau$ can be omitted.

6.8 DEFINITIONS: For $a\in A_\tau$ we define

$$a'=\begin{cases}a & \text{if } a\in A\\ \lambda & \text{if } a=\tau\end{cases}$$

If $a\cdot X_j$ is a summand of term T_i, then we say: $X_i\xrightarrow{a'}X_j$. $\longrightarrow\!\!\!\!\!\!\rightarrow$ is the transitive and reflexive closure of $\longrightarrow$: if $X_{j_0}\xrightarrow{a_1}X_{j_1}\xrightarrow{a_2}\cdots\xrightarrow{a_n}X_{j_n}$ and $\sigma=a_1{}^*a_2{}^*\cdots{}^*a_n$, then we say that $X_{j_0}\xrightarrow{\sigma}\!\!\!\!\!\!\rightarrow X_{j_n}$. The reflexivity is expressed by saying that $X_i\xrightarrow{\lambda}\!\!\!\!\!\!\rightarrow X_i$.

6.9 THEOREM. Let $E = \{E_j : j \in J\}$ be a guarded, strictly linear specification with solution p. Then

$$tr(p) = \{\sigma \mid \exists j \in J : X_{j_0} \xrightarrow{\sigma} X_j\}$$

In order to prove this theorem. we first prove some lemma's.

6.10 LEMMA. If the index set J of E is finite, and partially ordered by a relation $>_E$ such that j_0 is minimal and $\forall i,j \in J : X_i \in T_j \Rightarrow i >_E j$, then theorem 6.9 holds.

PROOF: We use induction on the number of elements of J.

CASE 1. $|J| = 1$ Because of the partial order on J, E_{j_0} has the form $X_{j_0} = \tau$ or $X_{j_0} = \delta$. This means $p = \tau$ or $p = \delta$. Hence $tr(p) = \{\lambda\} = \{\sigma \mid \exists j \in J : X_{j_0} \xrightarrow{\sigma} X_j\}$.

CASE 2. Suppose the lemma is proved for $|J| \leq n-1$. Consider the case $|J| = n$ $(n > 1)$. Because of the partial order on J, equation E_{j_0} must be of the form

$$X_{j_0} = \sum_{k=1}^{m} a_k \cdot X_{j_k}$$

Now we define for $1 \leq k \leq m : J_k = \{j \in J \mid \exists \sigma : X_{j_k} \xrightarrow{\sigma} X_j\}$. The designated element of J_k is j_k. The specification $E(k)$ is defined by $E(k) = \{E_j : j \in J_k\}$. $E(k)$ is strictly linear and guarded. Because $j_0 \notin J_k$ (partial order !), we have $|J_k| \leq n-1$. The restriction of the partial order $>_E$ on J to J_k gives us a partial order $>_{E(k)}$ on J_k such that $\forall i,j \in J_k : X_i \in T_j \Rightarrow i >_{E(k)} j$.
Let p_k be the unique solution of $E(k)$. This means that $p = \sum_{k=1}^{m} a_k \cdot p_k$. A simple induction argument gives us that p must be finite. Hence

$$tr(p) = tr(\sum_{k=1}^{m} a_k \cdot p_k) = \{\lambda\} \cup \bigcup_{k=1}^{m} a'_k * tr(p_k) = \quad \text{(induction)}$$

$$= \{\lambda\} \cup \bigcup_{k=1}^{m} a'_k * \{\sigma \mid \exists j \in J_k : X_{j_k} \xrightarrow{\sigma} X_j\} =$$

$$= \{\lambda\} \cup \bigcup_{k=1}^{m} \{a'_k * \sigma \mid \exists j \in J_k : X_{j_0} \xrightarrow{a'_k} X_{j_k} \xrightarrow{\sigma} X_j\} =$$

$$= \{\sigma \mid \exists j \in J : X_{j_0} g X_j\}$$

This finishes the proof of lemma 6.10.

6.11 DEFINITION. Let $E = \{E_j : j \in J\}$ be a strictly linear specification. For every $n \geq 1$ the specification $E^n = \{E^n_{j,m} : j \in J,\ m < n\} \cup \{E_\tau\}$ is defined as follows

(i) $T_\tau = \tau$
(ii) for $m < n-1$, $T^n_{j,m}$ is obtained from T_j by replacing each variable X_i in T_j by $X_{i,m+1}$ if the variable occurs guarded in T_j, and by $X_{i,m}$ otherwise.
(iii) for $m = n-1$, $T^n_{j,m}$ is obtained from T_j by replacing each variable X_i in T_j by X_τ if the variable occurs guarded in T_j, and by $X_{i,m}$ otherwise.

The index set J^n of E^n is $\{(j,m) \mid j \in J,\ m < n\} \cup \{\tau\}$; the special element of J^n is $(j_0,0)$. E^n is again a strictly linear recursive specification. When E is guarded E^n is guarded too and the transitive (but not reflexive) closure of $\longrightarrow$ is a partial order on J^n. If we throw away all the equations for unreachable variables the resulting index set is finite and $(j_0,0)$ is minimal.

6.12 EXAMPLE. The following example shows that in E^n the process specified by E is provided with a counter, which counts the number of 'real' steps, and cuts off the process after n 'real' steps.

$$E: \quad X = a \cdot X + \tau \cdot Y ; Y = b \cdot Y$$

$$E^2: \quad X_0 = a \cdot X_1 + \tau \cdot Y_0 ; Y_0 = b \cdot Y_1 ; X_\tau = \tau$$
$$X_1 = a \cdot X_\tau + \tau \cdot Y_1 ; Y_1 = b \cdot X_\tau$$

6.13 LEMMA. Let E be a guarded, strictly linear specification with solution p, and let $n \geqslant 1$. Then

$$E^n(\pi_n(p), -)$$

PROOF: By iterated application of lemma 1.5.5.

Now we are able to prove theorem 6.9.

PROOF: It is enough to show

$$\forall n \geqslant 1 \; \pi_n(tr(p)) = \pi_n(\{\sigma \mid \exists j \in J : X_{j_0} \xrightarrow{\sigma} X_j\})$$

Choose an arbitrary $n \geqslant 1$.

$$\pi_n(tr(p)) = tr(\pi_n(p)) = \text{(lemma 6.10 + lemma 6.13)}$$

$$= \{\sigma \mid \exists j \in J^n : X_{j_0,0} \xrightarrow{\sigma} X_j\} =$$

$$= \{\sigma \mid |\sigma| \leqslant n \wedge \exists j \in J : X_{j_0} \xrightarrow{\sigma} X_j\} =$$

$$= \pi_n(\{\sigma \mid \exists j \in J : X_{j_0} \xrightarrow{\sigma} X_j\})$$

Because n was chosen arbitrarily, this finishes the proof of theorem 6.9.

6.14 DEFINITION. Let $E = \{E_j : j \in J\}$ be a strictly linear recursive specification with solution p. Let $a \cdot X_j$ be a summand of T_i $(i,j \in J,\ a \in A_\tau)$. Let $q \in P \cup \{\epsilon\}$, $H \subseteq A$, and let p be observable in context $\partial_H(p \| q)$. The summand $a \cdot X_j$ is *redundant in context* $\partial_H(p \| q)$ iff

$$\{\sigma \mid X_{j_0} \xrightarrow{\sigma} X_i\}^* a' \cap \nu_p^{q,H} = \varnothing$$

E is *redundant in context* $\partial_H(p \| q)$ iff E has a summand which is redundant in this context.

6.15 Note. If a summand $a \cdot X_j$ of T_i is redundant in context $\partial_H(p \| q)$, we can depict this situation graphically as follows:

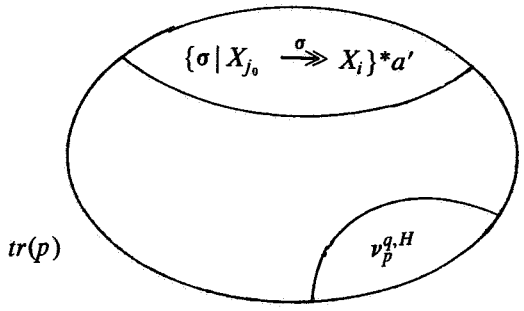


Fig. 22

This is because $\{\sigma \mid X_{j_0} \xrightarrow{\sigma} X_i\}^* a' \subseteq \{\sigma \mid \exists_{j \in J} : X_{j_0} \xrightarrow{\sigma} X_j\} = tr(p)$ (theorem 6.9), and $\nu_p^{q,H} \subseteq tr(p)$ (theorem 6.6).

6.16 THEOREM. Let E be a strictly linear guarded specification with solution p. Let q be ϵ or specifiable by a guarded specification without τ_I. Let $H \subseteq A$. Let p be observable in context $\partial_H(p \| q)$. Let $\overline{E}$ be a specification, obtained from E by omitting an arbitrary number of summands which are redundant in context $\partial_H(p \| q)$ (if in a term all summands are omitted, then this term becomes δ). Let $\overline{p}$ be the solution of $\overline{E}$ (because $\overline{E}$ is strictly linear and guarded, it has a unique solution). Then:

$$\partial_H(p \| q) = \partial_H(\overline{p} \| q)$$

The proof of this theorem is analogous to the proof of theorem 6.9, but a bit more complicated because there is also a q around.

6.17 LEMMA. If the index set J of E is finite, and partially ordered by a relation $>_E$ such that j_0 is minimal and $\forall i,j \in J : X_i \in T_j \Rightarrow i >_E j$, and if q is equal to ϵ, or is a finite process, then theorem 6.16 holds.

PROOF: We use simultaneous induction on the number of elements of J, and on the term representing q (element of $BT \cup \{\epsilon\}$), to prove the following three statements:
1) $\partial_H(p \| q) = \partial_H(\overline{p} \| q)$
2) $\partial_H(q \mathbin{\mathbb{L}} p) = \partial_H(q \mathbin{\mathbb{L}} \overline{p})$
3) $\partial_H(p \mid q) = \partial_H(\overline{p} \mid q)$
We only present the proof of statement 1. Statements 2 and 3 can be proved analogously.

CASE 1. $|J| = 1$
Because of the partial order on J, $T_{j_0} = \tau$ or $T_{j_0} = \delta$. Since E_{j_0} is the only equation of E, this means that E doesn't contain redundant summands, so there is nothing to prove.

CASE 2. Suppose the lemma is proved for $|J| \leq n-1$. Consider the case $|J| = n$ $(n > 1)$. Because of the partial order on J, equation E_{j_0} must be of the form

$$X_{j_0} = \sum_{k=1}^{m} a_k \cdot X_{j_k}$$

As in the proof of lemma 6.10, we define for $1 \leq k \leq m : J_k = \{j \in J \mid \exists \sigma : X_{j_k} \xrightarrow{\sigma} X_j\}$. The designated element of J_k is j_k. $E(k)$ is defined by $E(k) = \{E_j : j \in J_k\}$. $E(k)$ is a strictly linear and guarded

specification. Because $j_0 \notin J_k$, we have $|J_k| \leqslant n-1$. The restriction of the partial order $>_E$ on J to J_k gives us a partial order $>_{E(k)}$ on J_k such that $\forall i,j \in J_k : X_i \in T_j \Rightarrow i >_{E(k)} j$. Let p_k be the unique solution of $E(k)$. Hence $p = \sum_{k=1}^{m} a_k \cdot p_k$. Observe that p is finite.

The following five propositions allow us to use simultaneous induction.

PROPOSITION 1. If a summand of E is redundant in context $\partial_H(p \| \tau)$, then the summand is redundant in context $\partial_H(p \| \epsilon)$.

PROOF:

$$\begin{aligned} \nu_p^{\tau,H} &= tr(\partial_H(p \| \tau)) = tr(\partial_H(p \between \tau + \tau \between p + p|\tau)) = \\ &= tr(\tau \cdot \partial_H(p) + \partial_H(p \between \tau + p|\tau)) = \\ &= tr(\partial_H(p)) \cup tr(\partial_H(p \between \tau + p|\tau)) = \\ &= \nu_p^{\epsilon,H} \cup tr(\partial_H(p \between \tau + p|\tau)) \end{aligned}$$

So $\nu_p^{\epsilon,H} \subseteq \nu_p^{\tau,H}$. If a summand $a \cdot X_j$ of T_i is redundant in context $\partial_H(p \| \tau)$, then $\{\sigma \mid X_{j_0} \xrightarrow{\sigma} \gg X_i\}^* a' \cap \nu_p^{\tau,H} = \varnothing$. But then also $\{\sigma \mid X_{j_0} \xrightarrow{\sigma} \gg X_i\}^* a' \cap \nu_p^{\epsilon,H} = \varnothing$, which means that the summand is redundant in context $\partial_H(p \| \epsilon)$.

PROPOSITION 2. If a summand of E is redundant in a context $\partial_H(p \| ax)$ $(a \in A_\tau,\ x \in BT)$, and $a \notin H$, then the summand is redundant in context $\partial_H(p \| x)$.

PROOF: Analogous to the proof of proposition 1.

PROPOSITION 3. If a summand of E is redundant in a context $\partial_H(p \| (x+y))$ $(x,y \in BT)$, then the summand is redundant in context $\partial_H(p \| x)$.

PROOF: It is enough to show that $\nu_p^{x,H} \subseteq \nu_p^{x+y,H}$. Using induction, we can easily prove that $tr(\partial_H(p \| x)) \subseteq tr(\partial_H(p \| (x+y)))$. $\nu_p^{x,H}$ and $\nu_p^{x+y,H}$ can be obtained from $tr(\partial_H(p \| x))$ and $tr(\partial_H(p \| (x+y))$ by means of application of the operators t_H and λ_I. The inclusion relation however, is an invariant of these operators.

PROPOSITION 4. Let $1 \leqslant k \leqslant m$. Suppose $a_k \notin H$. If a summand of E, which is also a summand of $E(k)$, is redundant in a context $\partial_H(p \| q)$ $(q \in BT)$, then it is also redundant in context $\partial_H(p_k \| q)$.

PROOF: If a summand $a \cdot X_j$ of T_i is redundant in context $\partial_H(p \| q)$ then $\{\sigma \mid X_{j_0} \xrightarrow{\sigma} \gg X_i\}^* a' \cap \nu_p^{q,H} = \varnothing$. In particular $\{a'_k{}^* \sigma \mid X_{j_0} \xrightarrow{a'_k} X_{j_k} \xrightarrow{\sigma} \gg X_i\}^* a' \cap \nu_p^{q,H} = \varnothing$. A simple calculation gives us that $a'_k{}^* \nu_{p_k}^{q,H} \subseteq \nu_p^{q,H}$. Hence

$$a'_k{}^* \{\sigma \mid X_{j_k} \xrightarrow{\sigma} \gg X_i\}^* a' \cap a'_k{}^* \nu_{p_k}^{q,H} = \varnothing \Rightarrow \{\sigma \mid X_{j_k} \xrightarrow{\sigma} \gg X_i\}^* a' \cap \nu_{p_k}^{q,H} = \varnothing$$

This means that the summand is redundant in context $\partial_H(p_k \| q)$.

PROPOSITION 5. Let $1 \leqslant k \leqslant m$. If a summand of E, which is also a summand of $E(k)$, is redundant in a context $\partial_H(p \| ax)$ $(a \in A_\tau,\ x \in BT)$, and $a_k | a \notin H \cup \{\delta\}$, then the summand is redundant in context $\partial_H(p_k \| x)$.

PROOF: Analogous to the proof of proposition 4.

Note. Specification $\overline{E}$ is obtained out of specification E by omitting an arbitrary number of redundant summands. In the proof of case 2 we will make use of a specification $\tilde{E}$. Specification $\tilde{E}$ is like $\overline{E}$ obtained out of E, but now the summands $a \cdot X_j$ which were omitted in the construction of $\overline{E}$, are replaced by a summand $\delta \cdot X_j$. It will be clear that $\tilde{E}(\overline{p}, -)$.
In general $\tilde{E}$ is not strictly linear, but the strictly linear specification $\overline{E}$ can be obtained in a straightforward way from $\tilde{E}$. If $X_j = \sum_{k=1}^{m} a_k \cdot X_{j_k}$ is an equation which occurs in specification E, then $\tilde{E}$ contains a corresponding equation $X_j = \sum_{k=1}^{m} \overline{a}_k \cdot X_{j_k}$, with $\overline{a}_k = a_k$ or $\overline{a}_k = \delta$. In the last case the summand $a_k \cdot X_{j_k}$ of T_j is redundant. In the same way as the specifications $E(k)$ were introduced, we introduce the specifications $\tilde{E}(k)$. Let the solution of $\tilde{E}(k)$ be $\overline{p}_k$. We have

$$\overline{p} = \sum_{k=1}^{m} \overline{a}_k \cdot \overline{p}_k$$

Now we use induction on q:

CASE 2.1. $q = \epsilon$

$$\partial_H(p \| \epsilon) = \partial_H(p) = \partial_H(\sum_{k=1}^{m} a_k \cdot p_k) = \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k) =$$

$$= \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k \| \epsilon) =$$

(for each $1 \leqslant k \leqslant m$ there are two cases:

1. $a_k \in H$: $\partial_H(a_k) \cdot \partial_H(p_k \| \epsilon) = \delta = \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| \epsilon)$
2. $a_k \notin H$: the summand $\alpha_k \cdot X_{j_0}$ of X_{j_0} is not redundant, so $\partial_H(a_k) = \partial_H(\overline{a}_k)$. Proposition 4 allows us to use induction on the term $\partial_H(p_k \| \epsilon)$)

$$= \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| \epsilon) = \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k) = \partial_H(\sum_{k=1}^{m} \overline{a}_k \cdot \overline{p}_k) =$$

$$= \partial_H(\overline{p}) = \partial_H(\overline{p} \| \epsilon)$$

CASE 2.2. $q = \delta$

$$\partial_H(p \| \delta) = \partial_H(\sum_{k=1}^{m} a_k \cdot (p_k \| \delta) + \delta \cdot (\sum_{k=1}^{m} a_k \cdot p_k) + \sum_{k=1}^{m} (a_k | \delta) \cdot p_k) =$$

$$= \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k \| \delta) =$$

(same argument as in case 2.1)

$$= \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| \delta) = \cdots = \partial_H(\overline{p} \| \delta)$$

CASE 2.3. $q = \tau$

$$\partial_H(p \| \tau) = \partial_H(\sum_{k=1}^{m} a_k \cdot (p_k \| \tau) + \tau \cdot (p \| \epsilon) + \sum_{k=1}^{m} (a_k | \tau) \cdot p_k) =$$

$$= \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k \| \tau) + \tau \cdot \partial_H(p \| \epsilon) =$$

(use for the first term the same argument as in case 2.1, and use induction, which is allowed by proposition 1, for the second term)

$$= \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| \tau) + \tau \cdot \partial_H(\overline{p} \| \epsilon) = \cdots = \partial_H(\overline{p} \| \tau)$$

CASE 2.4. $q = ax$, $a \in A_\tau$

$$\partial_H(p \| ax) = \partial_H(\sum_{k=1}^{m} a_k \cdot (p_k \| ax) + a \cdot (x \| p) + \sum_{k=1}^{m} (a_k | a) \cdot (p_k \| x)) =$$

$$= \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k \| ax) + \partial_H(a) \cdot \partial_H(p \| x) + \sum_{k=1}^{m} \partial_H(a_k | a) \cdot \partial_H(p_k \| x) =$$

(as in case 2.1, we make for each term the distinction $a_k \in H$ or $a_k \notin H$, $a \in H$ or $a \notin H$, $(a_k | a) \in H \cup \{\delta\}$ or $(a_k | a) \notin H \cup \{\delta\}$. In the first case the step made is trivial, in the second case one of the propositions allows us to use induction)

$$= \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| ax) + \partial_H(a) \cdot \partial_H(\overline{p} \| x) + \sum_{k=1}^{m} \partial_H(\overline{a}_k | a) \cdot \partial_H(\overline{p}_k \| x) =$$

$$= \cdots = \partial_H(\overline{p} \| ax)$$

CASE 2.5. $q = x + y$

$$\partial_H(p \| (x + y)) = \partial_H(\sum_{k=1}^{m} a_k \cdot (p_k \| (x + y)) + x \mathbin{\llcorner} p + y \mathbin{\llcorner} p + p | x + p | y) =$$

$$= \sum_{k=1}^{m} \partial_H(a_k) \cdot \partial_H(p_k \| (x + y)) + \partial_H(x \mathbin{\llcorner} p) + \partial_H(y \mathbin{\llcorner} p) + \partial_H(p | x) + \partial_H(p | y) =$$

(case distinction and induction)

$$= \sum_{k=1}^{m} \partial_H(\overline{a}_k) \cdot \partial_H(\overline{p}_k \| (x + y)) + \partial_H(x \mathbin{\llcorner} \overline{p}) + \partial_H(y \mathbin{\llcorner} \overline{p}) + \partial_H(\overline{p} | x) + \partial_H(\overline{p} | y) =$$

$$= \cdots = \partial_H(\overline{p} \| (x + y))$$

This finishes the proof of lemma 6.17. Now we are able to prove theorem 6.16:

PROOF: Because p is specified by a guarded specification without τ_I, and $q = \epsilon$ or specifiable by a guarded specification without τ_I, $\partial_H(p \| q)$ is also specifiable by a guarded specification without τ_I. Hence, according to the Approximation Induction Principle (section 1.8), it is enough to prove (with $\pi_n(\epsilon) \equiv \epsilon$ for $n \geqslant 1$)

$$\forall n \geqslant 1 \quad \pi_n \circ \partial_H(p \| q) = \pi_n \circ \partial_H(\overline{p} \| q)$$

As shown by VAN GLABBEEK [9], this is equivalent to

$$\forall n \geqslant 1 \quad \pi_n \circ \partial_H(\pi_n(p) \| \pi_n(q)) = \pi_n \circ \partial_H(\pi_n(\overline{p}) \| \pi_n(q)).$$

Choose an arbitrary n.

CLAIM. $\overline{E}^n$ can be obtained from E^n by removing summands which are redundant in context $\partial_H(\pi_n(p) \| \pi_n(q))$.

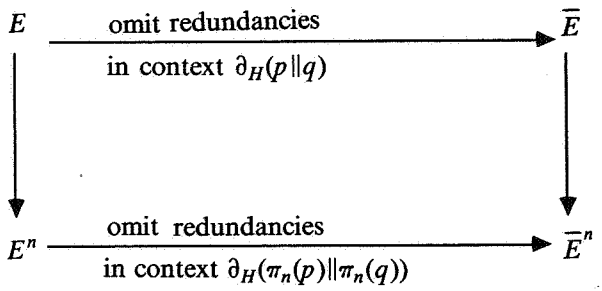


Fig. 23

PROOF: From the construction of E^n and $\overline{E}^n$ it is clear that $\overline{E}^n$ can be obtained from E^n by removing summands. Furthermore, these summands correspond to summands in E which are redundant in context $\partial_H(p\|q)$. So we have to prove that if a summand of E is redundant in context $\partial_H(p\|q)$, the corresponding summands of E^n are redundant in context $\partial_H(\pi_n(p)\|\pi_n(q))$.
Let $a \cdot X_j$ be a summand of term T_i of E, which is redundant in context $\partial_H(p\|q)$. So

$$\{\sigma|X_{j_0} \xrightarrow{\sigma}\!\!\!\gg X_i\}^* a' \cap \nu_p^{q,H} = \varnothing$$

But for all $m < n$:

$$\{\sigma|X_{j_0,0} \xrightarrow{\sigma}\!\!\!\gg X_{i,m}\} \subseteq \{\sigma|X_{j_0} \xrightarrow{\sigma}\!\!\!\gg X_i\}$$

Furthermore it is not very difficult to show that

$$\nu_{\pi_n(p)}^{\pi_n(q),H} \subseteq \nu_p^{q,H}$$

Hence for all $m < n$:

$$\{\sigma|X_{j_0,0} \xrightarrow{\sigma}\!\!\!\gg X_{i,m}\}^* a' \cap \nu_{\pi_n(p)}^{\pi_n(q),H} = \varnothing$$

This finishes the proof of the claim.

J_n is not necessarily finite, but $\{j \in J^n | \exists \sigma : X_{j_0,0} \xrightarrow{\sigma}\!\!\!\gg X_j\} \equiv \mathbb{J}^n$ is finite, and it is obvious that, without loss of generality, we can restrict E^n to the equations E_j for which $j \in \mathbb{J}^n$.

Because E is guarded, we can impose a partial order $>_{E^n}$ on the (new) index set $\mathbb{J}^n$ of E^n such that $(j_0,0)$ is minimal, and $\forall i,j \in \mathbb{J}^n : X_i \in T_j \Rightarrow i >_{E^n} j$.

Because $q = \epsilon$, or q is specifiable by a guarded specification without τ_I, $\pi_n(q)$ is finite or ϵ.

These facts, together with the claim, allow us to apply lemma 6.17, which gives us

$$\partial_H(\pi_n(p)\|\pi_n(q)) = \partial_H(\pi_n(\overline{p})\|\pi_n(q))$$

Hence

$$\pi_n \circ \partial_H(\pi_n(p) \| \pi_n(q)) = \pi_n \circ \partial_H(\pi_n(\overline{p}) \| \pi_n(q))$$

Because n was chosen arbitrarily this finishes the proof of theorem 6.16.

6.18 Note. We have defined the notion of redundancy only for strictly linear specifications. The definition of redundancy can however in an obvious way be extended to linear specifications.

A recursive specification is *linear* iff all terms are linear. If $\{X_j : j \in J\}$ is a set of variables then *linear terms* are inductively defined as follows

1. For $a \in A_{\tau,\delta}$, a is a linear term, and for $j \in J$, X_j is a linear term,
2. If T_1 and T_2 are linear terms, and $a \in A_\tau$, then so are $T_1 + T_2$ and $a \cdot T_1$.

Each linear specification corresponds in a natural way to a strictly linear specification. We give an example

$$\begin{array}{ll} X = abY & X = aX' \quad X' = bY \\ Y = c \quad \Rightarrow & Y = cX_\tau \quad X_\tau = \tau \end{array}$$

If in the strictly linear specification, which corresponds to a given linear specification, certain summands can be omitted in a certain context, we can translate this back in a straightforward way to the linear specification.

6.19 Redundancy in a context is undecidable

We show that it is undecidable (in general) whether or not a specification is redundant in a given context. We do this by showing that if it were decidable, we would have an algorithm for Post's Correspondence Problem (PCP). And such an algorithm can't be found because PCP itself is undecidable.

6.20 DEFINITION: An *instance* of PCP consist of two lists $A = x_1, \ldots, x_k$ and $B = y_1, \ldots, y_k$ of words over some alphabet Σ. This instance of PCP *has a solution* if there is any sequence of integers $i_1, \ldots, i_m$, with $m \geq 1$, such that $x_{i_1} * x_{i_2} * \ldots * x_{i_m} = y_{i_1} * y_{i_2} * \ldots * y_{i_m}$. The sequence $i_1, \ldots, i_m$ is a *solution* to this instance of PCP.

A proof that PCP is undecidable is presented in HOPCROFT & ULLMAN [10]. The following example can also be found in [10].

6.21 EXAMPLE: Let $\Sigma = \{0,1\}$. Let A and B be lists of three words each, as defined in Fig. 24. In this case PCP has a solution. Let $m = 4, i_1 = 2, i_2 = 1, i_3 = 1$ and $i_4 = 3$. Then $x_2x_1x_1x_3 = y_2y_1y_1y_3 =$ 101111110.

	List A	List B
i	x_i	y_i
1	1	111
2	10111	10
3	10	0

Fig. 24

6.22 THEOREM. Redundancy in a context in undecidable.

PROOF: Let $A = x_1, \ldots, x_k$ and $B = y_1, \ldots, y_k$, with x_i and y_i words over some alphabet Σ, be an instance of Post's Correspondence Problem. Consider the following network:

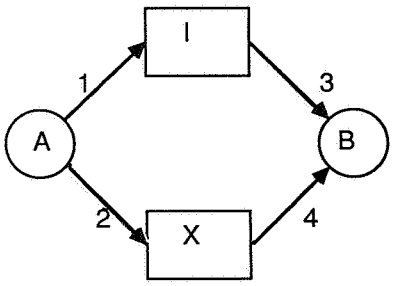


Fig. 25

The idea is the following: process A generates a sequence $i_1, \ldots, i_m$ (which is possibly a solution to the given instance of PCP). This sequence is sent into channel I, a FIFO queue with unbounded capacity. Each time an index i is sent into channel I, the corresponding word x_i is sent, character for character, into channel X, also a FIFO queue with unbounded capacity. After $i_1, \ldots, i_m$ is sent into channel I, and $x_{i_1}, \ldots, x_{i_m}$ into channel X, a special character $\perp$ (so $\perp \notin \Sigma$) is sent into channel I and into channel X.

On the other side of the channels process B checks if $x_{i_1} * \cdots * x_{i_m} = y_{i_1} * \cdots y_{i_m}$. If a deviation is detected process B deadlocks. If process B reads a $\perp$ at port 3 (this can only happen when an entire number of words is read at port 4), it tries to read a $\perp$ at port 4.

This can only happen when $x_{i_1} * \cdots * x_{i_m} = y_{i_1} * \cdots * y_{i_m}$ (PCP has a solution). But that is undecidable. This means that it is undecidable whether the action $r4(\perp)$ (reading a $\perp$-symbol at port 4) in the specification of process B is redundant in context $\partial_H(B \| (A \| I \| X))$.

We give recursive specifications of process A, B, I and X. Let $x_j = x_{j_1} \cdots x_{j_{m_j}}$ and $y_j = y_{j_1} \cdots y_{j_{n_j}}$ for $1 \leqslant j \leqslant k$ and let $D = \{1,...,k\} \cup \Sigma \cup \{\perp\}$. Then

$$A = \sum_{j=1}^{k} s1(j) \cdot S^j$$

$$S^j = s2(x_{j_1}) \cdot \cdots \cdot s2(x_{j_{m_j}}) \cdot T \quad (1 \leqslant j \leqslant k)$$

$$T = A + s1(\perp) \cdot s2(\perp)$$

$$B = \sum_{j=1}^{k} r3(j) \cdot U^j$$

$$U^j = r4(y_{j_1}) \cdot \cdots \cdot r4(y_{j_{n_j}}) \cdot V \quad (1 \leqslant j \leqslant k)$$

$$V = B + r3(\perp) \cdot r4(\perp)$$

$$I = I_\epsilon = \sum_{d \in D} r1(d) \cdot I_d$$

$$I_{\sigma * d} = s3(d) \cdot I_\sigma + \sum_{e \in D} r1(e) \cdot I_{e * \sigma * d} \quad d \in D,\ \sigma \in D^*$$

$$X = X_\epsilon = \sum_{d \in D} r2(d) \cdot X_d$$

$$X_{\sigma * d} = s4(d) \cdot X_\sigma + \sum_{e \in D} r2(e) \cdot X_{e * \sigma * d} \quad d \in D, \sigma \in D^*$$

Define communication by

$$st(d) \mid rt(d) = ct(d) \quad \text{for } t \in \{1,2,3,4\},\ d \in D$$

and let

$$H = \{st(d), rt(d) \mid t \in \{1,2,3,4\},\ d \in D\}$$

6.23 Proving redundancies

The proofs of the theorems presented on the previous pages were long and rather technical. However, in their application, the theorems provide us with a simple and powerful tool, which allows us to prove statements about the trace set of a concurrent system, and to detect and remove redundancies.

The basic situation we want to analyze is a system which consists of the encapsulated merge of a number of components:

$$P = \partial_H(P_1 \| \cdots \| P_n)$$

A reasonable assumption is that each component is observable in context with the others. As definition 6.14 shows, it is necessary, in order to prove redundancies, to have information about

$$tr(P)$$

However, if the number of components of P is large calculating $tr(P)$ becomes a tremendous job, which is even beyond the reach of a supercomputer. It is far more easier to calculate the trace sets of a component (use theorem 6.9):

$$tr(P_1)$$

Theorem 6.6 now gives us some information about $tr(P)$:

$$\nu_{P_1}^{(P_2 \| \cdots \| P_n),H} \subseteq tr(P_1)$$

since $\nu_{P_1}^{(P_2 \| \cdots \| P_n),H}$ is obtained from $tr(P)$ by means of renamings. If, for example, in every element of $tr(P_1)$ every b is always preceded by an a, theorem 6.6 yields that in $tr(P)$ the renamed version of b is always precede by the renamed version of a. We use this type of reasoning in the proof of the following lemma. (see sections 5.9 - 5.16 for the specifications of the various components)

6.24 LEMMA. Let *Obsw* be specified as follows:

$$Obsw = \partial_{H4}(IMP_A \| T_A \| K_1 \| R_B \| IMP_B \| T_B \| L_1 \| R_A)$$

(So $OBSW = \tau_{I2}(Obsw)$). Then

$$\forall d \in D\ \forall p \in B: \quad c2(d,p,p) \notin tr(Obsw)$$

PROOF: Suppose

$$\exists \sigma \in A^*\ \exists d \in D\ \exists p \in B : \sigma^* c2(d,p,p) \in tr(Obsw)$$

Without loss of generality we may assume $\forall e \in D\ \forall q \in B : c2(e,q,q) \notin \sigma$.
If we look at the specification of IMP_A and apply theorem 6.6 this gives us

$$\exists \sigma_1, \sigma'_1 \in A^*\ \exists d_1 \in D\ \exists p_1 \in B : \sigma = \sigma_1{}^* c10(d_1, 1-p_1, p_1)^* \sigma'_1$$

Now look at the specification of R_A and apply again theorem 6.6

$$\exists \sigma_2, \sigma'_2 \in A^* : \sigma_1 = \sigma_2{}^* c3(d_1, 1-p_1, p_1)^* \sigma'_2$$

We iterate this procedure a number of times; look at the specification of L_1:

$$\exists \sigma_3, \sigma'_3 \in A^* : \sigma_2 = \sigma_3{}^* c6(d_1, 1-p_1, p_1)^* \sigma'_3$$

We walk backward through IMP_B:

$$\exists \sigma_4, \sigma'_4 \in A^*\ \exists d_2 \in D\ \exists p_2 \in B : \sigma_3 = \sigma_4{}^* c12(d_2, p_2, p_2)^* \sigma'_4$$

from port 12 backward to port 7 through R_B,

$$\exists \sigma_5, \sigma'_5 \in A^* : \sigma_4 = \sigma_5{}^* c7(d_2, p_2, p_2)^* \sigma'_5$$

and from port 7 via channel K_1 to port 2, the starting point of our excursion:

$$\exists \sigma_6, \sigma'_6 \in A^* : \sigma_5 = \sigma_6{}^* c2(d_2, p_2, p_2)^* \sigma'_6$$

And since σ_5 is a prefix of σ this gives the desired contradiction.

6.25 It will be clear that we can generalize the result of lemma 6.24: if a frame is communicated at port 2, 7 or 12 it is of the form $(d, 1-p, p)$, and if it is communicated at ports 6, 3 or 10 it is of the form (d,p,p). In other words: the first control bit *determines* the second one. This means that there are a number of redundancies in the specifications of the components. If we apply theorem 6.16 to remove these redundancies, and omit the second control bit out of the communication actions (an abbreviation) this gives us the following theorem:

6.26 THEOREM. The following specification of OBSW is equivalent to the old one ($d \in D$; $p \in B$):

$$IMP_A = \tau \cdot \sum_{d \in D} r1(d) \cdot SF_A^{d0}$$

$$SF_A^{dp} = s2(d,p) \cdot WS_A^{dp}$$

$$WS_A^{dp} = (r9(to) + r9(ce)) \cdot SF_A^{dp} + r9(fa) \cdot GF_A^{dp}$$

$$GF_A^{dp} = \sum_{e \in D} r10(e, 1-p) \cdot SF_A^{dp} + \\ + \sum_{e \in D} r10(e,p) \cdot s4(e) \cdot \sum_{f \in D} r1(f) \cdot SF_A^{f(1-p)}$$

$$IMP_B = \tau \cdot \sum_{d \in D} r5(d) \cdot WF_B^{d0}$$

$$WF_B^{d0} = r11(ce) \cdot WF_B^{d0} + r11(fa) \cdot \sum_{e \in D} r12(e0) \cdot s8(e) \cdot SF_B^{d0}$$

$$SF_B^{dp} = s6(d,p) \cdot WS_B^{dp}$$

$$WS_B^{dp} = (r11(to) + r11(ce)) \cdot SF_B^{dp} + r11(fa) \cdot GF_B^{dp}$$

$$GF_B^{dp} = \sum_{e \in D} r12(e,p) \cdot SF_B^{dp} + \\ + \sum_{e \in D} r12(e, 1-p) \cdot s8(e) \cdot \sum_{f \in D} r5(f) \cdot SF_B^{f(1-p)}$$

$$T_A = s9(to) \cdot T_A$$

$$T_B = s12(to) \cdot T_B$$

$$R_A = \sum_{f \in DB} r3(f) \cdot s9(fa) \cdot s10(f) \cdot R_A + r3(ce) \cdot s9(ce) \cdot R_A$$

$$R_B = \sum_{f \in DB} r7(f) \cdot s11(fa) \cdot s12(f) \cdot R_B + r7(ce) \cdot s11(ce) \cdot R_B$$

$$K_1 = K_1^{\varepsilon} = \sum_{f \in DB} r2(f) \cdot K_1^f \qquad f \in DB; \sigma \in (DB)^*$$

$$K_1^{\sigma * f} = (s7(f) + s7(ce) + \tau) \cdot K_1^{\sigma} + \sum_{g \in DB} r2(g) \cdot K_1^{g * \sigma * f}$$

$$L_1 = L_1^{\varepsilon} = \sum_{f \in DB} r6(f) \cdot L_1^f \qquad f \in DB; \sigma \in (DB)^*$$

$$L_1^{\sigma * f} = (s3(f) + s3(ce) + \tau) \cdot L_1^{\sigma} + \sum_{g \in DB} r6(g) \cdot L_1^{g * \sigma * f}$$

$$H = \{st(f), rt(f) | t \in \{2,3,6,7,9,10,11,12\}, f \in DB \cup \{ce,to,fa\}\}$$

$$I = \{ct(f) | t \in \{2,3,6,7,10,11,12\}, f \in DB \cup \{ce,to,fa\}\}$$

$$OBSW = \tau_I \circ \partial_H(R_A \| IMP_A \| T_A \| K_1 \| R_B \| IMP_B \| T_B \| L_1)$$

6.27 REMARK. The redundancy in the specification of the OBSW-protocol can be retraced to a redundancy in the computer progam which is presented in TANENBAUM [12]. The presence of this redundancy can be explained as follows: In his book Tanenbaum presents the One Bit Sliding Window protocol as an introduction to the general sliding window protocols (in which the window size is larger than 1). If the window size is larger than 1, the redundancy does *not* occur (the value of the field *s.seq* does *not* determine the value of the field *s.ack*).

This means that the One Bit Sliding Window Protocol has a behaviour which is essentially different from the behaviour of Sliding Window protocols with a larger window size.

7.1 Local Replacement. For the verification of the OBSW-protocol we will make extensive use of a technique which we shall call 'local replacement'. In fact we used the technique already in some of the preliminary calculations for the OBSW-protocol (notably lemma 5.15 and theorem 6.26). In this section however we will make it explicit.

The basic situation we have to deal with is the abstracted and encapsulated merge of a number of components:

$$\tau_I \circ \partial_H(X_1 \| \cdots \| X_n)$$

In general the complexity of this system is immense. It is difficult to verify statements about the system. Now the basic idea behind the local replacement is that one proves that one can replace certain components by new components, without changing the properties of the system as a whole. A replacement makes sense if the complexity of the resulting system is less than the complexity of the original system. After the first replacement, for example of X_1 and X_2 by X'_1 and X'_2, we can perform a second one, for example the replacement of X'_2 and X_3 by X''_2 (the number of components can change). Thus we use local replacement to reduce stepwise the complexity of the system as a whole.

Of course the succes of the local replacement technique depends on the ability to find succesful replacements. By now two mechanisms are available.

7.1.1 Redundancy. In section 6 we saw how the trace sets of the components give us information about the trace set of the system

$$\partial_H(X_1 \| \cdots \| X_n)$$

Further we saw how this information can be used to prove redundancies. And since redundancies can be removed (theorem 6.16), this gives us a mechanism to accomplish replacements:

$$\tau_I \circ \partial_H(X_1 \| (X_2 \| \cdots \| X_n)) \to \tau_I \circ \partial_H(X'_1 \| (X_2 \| \cdots \| X_n))$$

7.1.2 Conditional Axioms. The conditional axioms (in particular CA1 and CA2) allow us to 'pull τ_I- and ∂_H-operators through a merge'. This can be used to accomplish replacements in the following way:
(step 1)

$$\tau_I \circ \partial_H(X_1 \| \cdots \| X_5) = \text{(conditional axioms)}$$
$$= \tau_I \circ \partial_H(\tau_{I'} \circ \partial_{H'}(X_1 \| X_2) \| X_3 \| X_4 \| X_5)$$

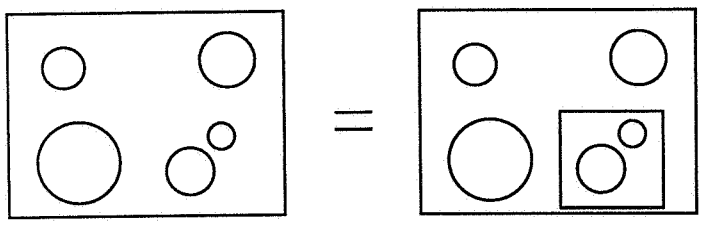


Fig. 26

(step 2)

$$\tau_{I'} \circ \partial_{H'}(X_1 \| X_2) = \tau_{I'} \circ \partial_{H'}(X'_1 \| X'_2)$$

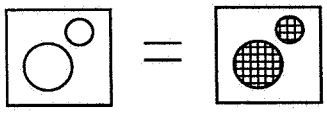


Fig. 27

(step 3)

$$\tau_I \circ \partial_H(\tau_{I'} \circ \partial_{H'}(X'_1 \| X'_2) \| X_3 \| X_4 \| X_5) = \text{(conditional axioms)}$$
$$= \tau_I \circ \partial_H(X'_1 \| X'_2 \| X_3 \| X_4 \| X_5)$$

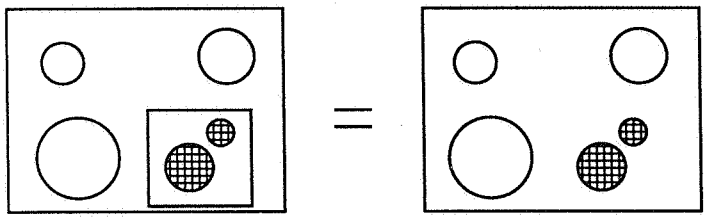


Fig. 28

So we will show by means of the local replacement technique that the OBSW-protocol is a valid communication protocol. The first thing we do (lemma 7.2) is that we give the receiver a 'memory' to distinguish a frame that it is seeing for the first time from a retransmission. Further we make the observation that it does not matter if the receiver loses a frame now and then, since the communication channel can lose frames too.

7.2 LEMMA. Let R_{A1} be the specified as follows ($p \in B$):

$$R_{A1} = R^0_{A1}$$

$$R^p_{A1} = \sum_{d \in D} r3(d,p) \cdot s9(fa) \cdot s10(d,p) \cdot R^{1-p}_{A1} +$$

$$+ \sum_{d \in D} r3(d, 1-p) \cdot s9(fa) \cdot s10(d, 1-p) \cdot R^p_{A1} +$$

$$+ \sum_{d \in D} r3(d, 1-p) \cdot R^p_{A1} +$$

$$+ r3(ce) \cdot s9(ce) \cdot R^p_{A1} + r3(ce) \cdot R^p_{A1}$$

Let $H1$ and $I1$ be the following sets:

$$H1 = \{r3(f), s3(f) | f \in DB \cup \{ce\}\}$$

$$I1 = \{c3(f) | f \in DB \cup \{ce\}\}$$

then

$$\tau_{I1} \circ \partial_{H1}(L_1 \| R_A) = \tau_{I1} \circ \partial_{H1}(L_1 \| R_{A1})$$

PROOF: Straightforward.

7.3 COROLLARY

$$OBSW = \tau_I \circ \partial_H(R_{A1} \| IMP_A \| T_A \| K_1 \| R_B \| IMP_B \| T_B \| L_1)$$

PROOF: Local replacement.

The basic idea behind the following lemma is that :

(1) If R_{A1} sends a checksum error message or an old frame to IMP_A, the only result of this is that IMP_A sends a copy of the last sent frame to IMP_B.
(2) A timeout-action causes IMP_A to do the same.
(3) If R_{A1} reads a checksum error or an old frame at port 3, this does not necessarily mean that thereafter this information is passed to IMP_A.
(4) If IMP_A sends a copy of the last sent frame to IMP_B it is (after appropriate abstraction) unclear if this was caused by a timeout or by the arrival at R_{A1} of a checksum error message or of an old frame.
(5) It is allowed to change R_{A1} in such a way that it does not send checksum error messages or old frames to IMP_A.

7.4 LEMMA. Let R_{A2} be specified as follows ($p \in B$):

$$R_{A2} = R^0_{A2}$$

$$R^p_{A2} = \sum_{d \in D} r3(d,p) \cdot s9(fa) \cdot s10(d,p) \cdot R^{1-p}_{A2} + \\ + \sum_{d \in D} r3(d,1-p) \cdot R^p_{A2} + r3(ce) \cdot R^p_{A2}$$

Let $H2$ and $I2$ be the following sets

$$H2 = \{st(f), rt(f) | t \in \{9,10\}; f \in DB \cup \{ce,to,fa\}\}$$
$$I2 = \{ct(f) | t \in \{9,10\}; f \in DB \cup \{ce,to,fa\}\}$$

then

$$\tau_{I2} \circ \partial_{H2}(R_{A1} \| IMP_A \| T_A) = \tau_{I2} \circ \partial_{H2}(R_{A2} \| IMP_A \| T_A)$$

PROOF: Straightforward (in essence, an application of axiom T3).

Since R_{A2} never sends an old frame or a checksum error message to IMP_A, a number of summands in the specification of IMP_A are redundant. Application of theorem 6.16, together with the rather trivial elimination of the timer, gives us the following lemma:

7.5 LEMMA. Let IMP_{A1} be given by the following equations ($d \in D, p \in B$):

$$IMP_{A1} = \tau \cdot \sum_{d \in D} r1(d) \cdot SF^{d0}_{A1}$$

$$SF^{dp}_{A1} = s2(d,p) \cdot WS^{dp}_{A1}$$

$$WS^{dp}_{A1} = \tau \cdot SF^{dp}_{A1} + r9(fa) \cdot GF^{dp}_{A1}$$

$$GF^{dp}_{A1} = \sum_{e \in D} r10(e,p) \cdot s4(e) \cdot \sum_{f \in D} r1(f) \cdot SF^{f(1-p)}_{A1}$$

then

$$\tau_{I2} \circ \partial_{H2}(R_{A2} \| IMP_A \| T_A) = \tau_{I2} \circ \partial_{H2}(R_{A2} \| IMP_{A1})$$

PROOF:-

7.6 COROLLARY.

$$OBSW = \tau_I \circ \partial_H(R_{A2} \| IMP_{A1} \| K_1 \| R_B \| IMP_B \| T_B \| L_1)$$

PROOF: Local replacement.

The sending behaviour of IMP_{A1} is, to a large extent, independent of the incoming messages at port 3. This allows us to split IMP_{A1} up in two parts: a sender-component and an IMP-component. To this purpose we add to the parameters of atomic actions the parameter ss (= start sending), and

extend the communication function by

$$st(ss)|rt(ss) = ct(ss) \quad t \in \{1,2,\ldots,12\}$$

Now we can easily prove the following lemma:

7.7 LEMMA. Let IMP_{A2} and S_A be given by the following equations ($p \in B, d \in D$):

$$IMP_{A2} = s9(ss) \cdot \sum_{d \in D, p \in B} r10(d,p) \cdot s4(d) \cdot IMP_{A2}$$

$S_A = S_A^0$	"sender"
$S_A^p = r9(ss) \cdot SR_A^p$	
$SR_A^p = \sum_{d \in D} r1(d) \cdot SS_A^{dp}$	"sender reads"
$SS_A^{dp} = s2(d,p) \cdot SW_A^{dp}$	"sender sends"
$SW_A^{dp} = \tau \cdot SS_A^{dp} + r9(fa) \cdot S_A^{1-p}$	"sender waits"

Let $H3$ and $I3$ be the following sets

$$H3 = \{st(f),rt(f) | t \in \{9,10\}; f \in DB \cup \{ce,fa,ss\}\}$$
$$I3 = \{ct(f) | t \in \{9,10\}; f \in DB \cup \{ce,fa,ss\}\}$$

then

$$T_{I3} \circ \partial_{H3}(R_{A2} \| IMP_{A1}) = \tau_{I3} \circ \partial_{H3}(R_{A2} \| IMP_{A2} \| S_A)$$

PROOF:-

The receiver R_{A2} throws away all the old and damaged frames. The following lemma states that this job can be performed also by channel L_1.

7.8 LEMMA. Let L_2 and R_{A3} be specified as follows ($\sigma \in (DB)^*, p \in B, d \in D$):

$$
\begin{aligned}
L_2 &= L_2^{\varepsilon,0} \\
L_2^{\varepsilon,p} &= \sum_{f \in DB} r6(f) \cdot L_2^{f,p} \\
L_2^{\sigma^*[d,p],p} &= s3(d,p) \cdot L_2^{\sigma,1-p} + \tau \cdot L_2^{\sigma,p} + \sum_{g \in BD} r6(g) \cdot L_2^{g^*\sigma^*[d,p],p} \\
L_2^{\sigma^*[d,1-p],p} &= \tau \cdot L_2^{\sigma,p} + \sum_{g \in BD} r6(g) \cdot L_2^{g^*\sigma^*[d,1-p],p}
\end{aligned}
$$

$$
R_{A3} = \sum_{f \in DB} r3(f) \cdot s9(fa) \cdot s10(f) \cdot R_{A3}
$$

Let $H4$ and $I4$ be the following sets

$$H4 = \{r3(f), s3(f) | f \in DB \cup \{ce\}\}$$
$$I4 = \{c3(f) | f \in DB \cup \{ce\}\}$$

then

$$\tau_{I4} \circ \partial_{H4}(L_1 \| R_{A2}) = \tau_{I4} \circ \partial_{H4}(L_2 \| R_{A3})$$

PROOF: Straightforward.

It will be clear that we can derive analogous versions of lemmas 7.2 - 7.8 for channel K_1 and the components of system B. The only real difference is caused by the initialization phase of IMP_B. We summarize all results in one lemma:

7.9 LEMMA. Let K_2, IMP_{B2}, S_B and R_{B3} be given by the following equations ($p \in B, d \in D$):

$$
\begin{aligned}
K_2 &= K_2^{\varepsilon,0} \\
K_2^{\varepsilon,p} &= \sum_{f \in DB} r2(f) \cdot K_2^{f,p} \\
K_2^{\sigma^*[d,p],p} &= s7(d,p) \cdot K_2^{\sigma,p} + \tau \cdot K_2^{\sigma,1-p} + \sum_{g \in BD} r2(g) \cdot K_2^{g^*\sigma^*[d,p],p} \\
K_2^{\sigma^*[d,1-p],p} &= \tau \cdot K_2^{\sigma,p} + \sum_{g \in BD} r2(g) \cdot K_2^{g^*\sigma^*[d,1-p],p}
\end{aligned}
$$

$$IMP_{B2} = s11(ss) \cdot \sum_{d \in D, p \in B} r12(d,p) \cdot s8(d) \cdot IMP_{B2}$$

$$\begin{aligned}
S_B &= r11(ss) \cdot \sum_{d \in D} r5(d) \cdot r11(fa) \cdot r11(ss) \cdot SS_B^{d0} \\
S_B^p &= r11(ss) \cdot SR_B^p \\
SR_B^p &= \sum_{d \in D} r5(d) \cdot SS_B^{dp} \\
SS_B^{dp} &= s6(d,p) \cdot SW_B^{dp} \\
SW_B^{dp} &= \tau \cdot SS_B^{dp} + r11(fa) \cdot S_B^{1-p}
\end{aligned}$$

$$R_{B3} = \sum_{f \in BD} r7(f) \cdot s11(fa) \cdot s12(f) \cdot R_{B3}$$

Let $H5$ and $I5$ be the following sets

$$H5 = \{st(f), rt(f) | t \in \{2,3,6,7,9,10,11,12\}, f \in DB \cup \{ce,fa,ss\}\}$$
$$I5 = \{ct(f) | t \in \{2,3,6,7,9,10,11,12\}, f \in DB \cup \{ce,fa,ss\}\}$$

then

$$OBSW = \tau_{I5} \circ \partial_{H5}(IMP_{A2} \| R_{A3} \| S_A \| K_2 \| IMP_{B2} \| R_{B3} \| S_B \| L_2)$$

PROOF: Straightforward.

This lemma alters the overall picture of our system drastically. The protocol we are considering now is completely different form the original one. Only after abstraction they are the same. Fig. 29 depicts the new version of Fig. 20.

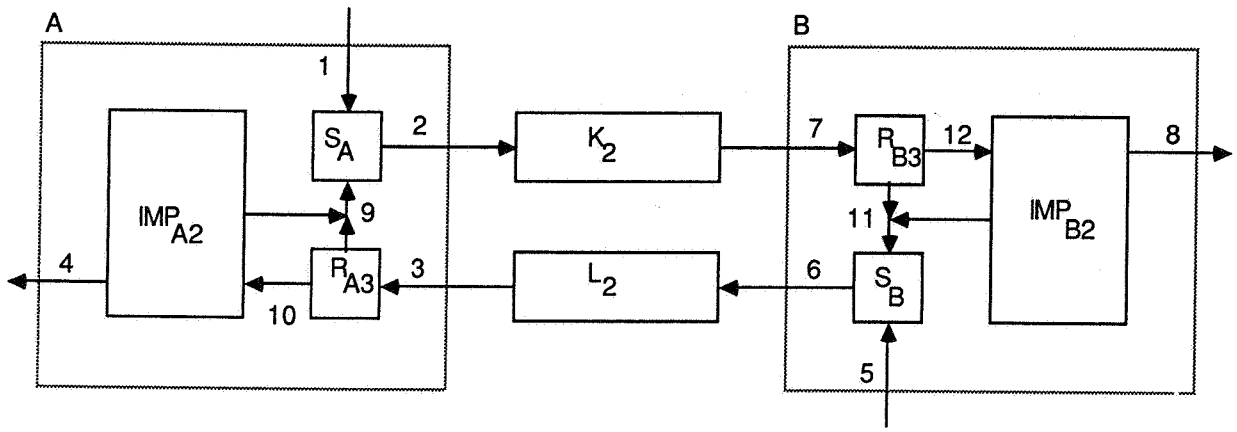


Fig. 29

At first sight we have not made any progress at all, since the beginning of this section. The specification of the OBSW-protocol we have now is longer than the one in theorem 6.26. However, we have been able to separate the sending and the receiving behaviour of systems A and B, and this makes it possible to replace S_A and K_2 on the one hand, and S_B and L_2 on the other, by very simple specifications. But before we can do this, still some work has to be done.

The specification of channel K_2 looks complicated, but has some nice properties. In lemma 7.12 we will prove that for $d \in D$ and $p \in B$:

$$K_2^{[d,1-p],p} = \tau \cdot K_2^{\epsilon,p}$$

Before we do this, we first give two auxiliary lemmas.

The first lemma says that channel K_2, in state p and with content σ ($K_2^{\sigma,p}$), can be thought of as the merge of two processes: a read-process K_r^{ϵ}, and a send-process $K_s^{\sigma,p}$. The read-process reads new data at port 2, the send-process sends the data of σ at port 7. When the send-process is finished, and is in state $K_s^{\epsilon,q}$, it gives a signal $s13(q)$ to the read-process. After this signal is received by the read-process (which is in state K_r^{σ}), the read-process becomes a normal channel again ($K_2^{\sigma',q}$).
We add to the alphabet of atomic actions:

$$r13(p), s13(p) \text{ and } c13(p) \quad (p \in B)$$

and extend the communication function by

$$r13(p) | s13(p) = c13(p) \quad (p \in B)$$

7.10 LEMMA. Let, for $\sigma\in(BD)^*, d\in D, p\in B$:

K_r^ϵ	$= \sum_{g\in BD} r2(g)\cdot K_r^g$
$K_r^{\sigma^* f}$	$= \sum_{g\in BD} r2(g)\cdot K_r^{g^*\sigma^* f} + \sum_{q\in B} r13(q)\cdot K_2^{\sigma^* f,p}$
$K_s^{\epsilon,p}$	$= s13(p)$
$K_s^{\sigma^*[d,p],p}$	$= s7(d,p)\cdot K_s^{\sigma,1-p} + \tau\cdot K_s^{\sigma,p}$
$K_s^{\sigma^*[d,1-p],p}$	$= \tau\cdot K_s^{\sigma,p}$

let $H6$ and $I6$ be the following sets

$$H6 = \{r13(p), s13(p) | p\in B\}$$

$$I6 = \{c13(p) | p\in B\}$$

then for $\sigma\in(BD)^*$ and $p\in B$

$$K_2^{\sigma,p} = \tau_{I6}\circ\partial_{H6}(K_r^\epsilon \| K_s^{\sigma,p})$$

PROOF: Straightforward.

7.11 LEMMA. If $x,y\in P$ are specifiable by a guarded specification without τ_I operator, then

$$x\|\tau y = \tau\cdot(x\|y)$$

PROOF: Use simultaneous induction on the structure of the basic terms representing x and y to prove the finite case. The infinite case can be dealt with by application of AIP, and by the use of the identity

$$\pi_n(x\|\tau y) = \pi_n(\pi_n(x)\|\pi_n(\tau y)) \quad (n\geqslant 1)$$

(VAN GLABBEEK [9]).

7.12 LEMMA. For $d,e\in D$ and $p\in B$ we have the following identities

(i)

$$K_2^{[d,1-p],p} = \tau\cdot K_2^{\epsilon,p}$$

(ii)

$$K_2^{[d,p]^*[d,p],p} = \tau\cdot K_2^{[d,p],p}$$

PROOF: The following identities hold for the send-process:

(1)

$$K_s^{[d,1-p],p} = \tau\cdot K_s^{\epsilon,p}$$

(2)

$$K_s^{[d,p]^*[d,p],p} = s7(d,p)\cdot K_s^{[d,p],1-p} + \tau\cdot K_s^{[d,p],p} =$$

$$= s7(d,p)\cdot K_s^{\epsilon,1-p} + \tau\cdot(s7(d,p)\cdot K_s^{\epsilon,1-p} + \tau\cdot K_s^{\epsilon,p}) =$$

$$= \tau \cdot (s7(d,p) \cdot K_s^{\epsilon,1-p} + \tau \cdot K_s^{\epsilon,P}) =$$
$$= \tau \cdot K_s^{[d,p],p}$$

The identities (i) and (ii) follow directly from identities (1) and (2) by application of lemmas 7.10 and 7.11.

Define *obsw* as follows:

$$obsw = \partial_{H5}(IMP_{A2} \| R_{A3} \| S_A \| K_2 \| IMP_{B2} \| R_{B3} \| S_B \| L_2)$$

(so we have $\tau_{H5}(obsw) = OBSW$).
The following lemma gives a property of $tr(obsw)$ which will be used to prove a redundancy. The lemma states that a frame-arrival message is not passed to the sender before the frame actually has arrived.

7.13 Note. We use $|a \in \sigma|$ as a notation for $|\lambda_{A-\{a\}}(\sigma)|$.

7.14 LEMMA. $\forall \sigma \in tr(obsw)$

$$|c9(fa) \in \sigma| \leqslant \sum_{f \in DB} |c7(f) \in \sigma| \leqslant |c9(fa) \in \sigma| + 1$$

PROOF: (sketch) Each of the components of *obsw* is observable in context with the others. This means we can apply theorem 6.6 for each of the components.
1.

$$|c9(fa) \in \sigma| \leqslant \sum_{f \in BD} |c3(f) \in \sigma|$$

(look at specification of component R_{A3})
2.

$$\sum_{f \in DB} |c3(f) \in \sigma| \leqslant |c11(fa) \in \sigma|$$

(look at specification of components L_2 and S_B. The sequence number of successive frames sent by channel L_2 at port 3 are different. This means that the number of frames sent at port 3 is less than the number of times the sequence number of the frames read at port 6 changed. But if we look at the specification of S_B, we see that this last number is less than the number of frame-arrival messages at port 11)
3.

$$|c11(fa) \in \sigma| \leqslant \sum_{f \in DB} |c7(f) \in \sigma|$$

(look at specification of component R_{B3})
4.

$$\sum_{f \in DB} |s7(f) \in \sigma| \leqslant |c9(fa) \in \sigma| + 1$$

(look at specification of components K_2 and S_{A3}; same argument as in 2)
The lemma follows from the combination of 1, 2, 3 and 4.

In the proof of the following lemma we use lemmas 7.11, 7.12 and 7.14.

7.15 LEMMA. Let X be specified as follows ($d \in D, p \in B$):

$$
\begin{aligned}
X &= X_1^0 \\
X_1^p &= r9(ss) \cdot X_2^p \\
X_2^p &= \sum_{d \in D} r1(d) \cdot X_3^{dp} \\
X_3^{dp} &= c2(d,p) \cdot X_4^{dp} \\
X_4^{dp} &= \tau \cdot X_5^{dp} + s7(d,p) \cdot X_6^{dp} + \tau \cdot X_7^{dp} \\
X_5^{dp} &= c2(d,p) \cdot X_4^{dp} + s7(d,p) \cdot X_8^{dp} + \tau \cdot X_3^{dp} \\
X_6^{dp} &= \tau \cdot X_8^{dp} + r9(fa) \cdot X_1^{1-p} \\
X_7^{dp} &= \tau \cdot X_3^{dp} \\
X_8^{dp} &= c2(d,p) \cdot X_6^{dp}
\end{aligned}
$$

then

$$obsw = \partial_{H5}(IMP_{A2} \| R_{A3} \| X \| IMP_{B2} \| R_{B3} \| S_B \| L_2)$$

PROOF: Let $H7 = \{r2(f), s2(f) | f \in DB\}$ then

$$obsw = \partial_{H5}(IMP_{A2} \| R_{A3} \| \partial_{H7}(S_A \| K_2) \| IMP_{B2} \| R_{B3} \| S_B \| L_2)$$

Now look at the equations below (the numbers of the equations correspond to the subscripts of variables in the specification of X). Observe that if we omit the summands in blocks, we have

$$X = \partial_{H7}(S_A \| K_2)$$

by application of RSP. See Fig. 30 for the state-transition diagram of X.

$$\partial_{H7}(S_A \| K_2) = \partial_{H7}(S_A^0 \| K_2^{\epsilon 0}) \tag{-}$$

$$\partial_{H7}(S_A^p \| K_2^{\epsilon p}) = r9(ss) \cdot \partial_{H7}(SR_A^p \| K_2^{\epsilon p}) \tag{1}$$

$$\partial_{H7}(SR_A^p \| K_2^{\epsilon p}) = \sum_{d \in D} r1(d) \cdot \partial_{H7}(SS_A^{dp} \| K_2^{\epsilon p}) \tag{2}$$

$$\partial_{H7}(SS_A^{dp} \| K_2^{\epsilon p}) = c2(d,p) \cdot \partial_{H7}(SW_A^{dp} \| K_2^{[d,p],p}) \tag{3}$$

$$\partial_{H7}(SW_A^{dp} \| K_2^{[d,p],p} = \tau \cdot \partial_{H7}(SS_A^{dp} \| K_2^{[d,p],p} + \tag{4}$$

$$\boxed{+ r9(fa) \cdot \partial_{H7}(S_A^{1-p} \| K_2^{[d,p],p}) +}$$

$$+ s7(d,p) \cdot \partial_{H7}(SW_A^{dp} \| K_2^{\epsilon 1-p}) +$$

$$+ \tau \cdot \partial_{H7}(SW_A^{dp} \| K_2^{\epsilon p})$$

$$\partial_{H7}(SS_A^{dp} \| K_2^{[d,p],p}) = c2(d,p) \cdot \partial_{H7}(SW_A^{dp} \| K_2^{[d,p]*[d,p],p}) + \tag{5}$$

$$+ s7(d,p)\cdot\partial_{H7}(SS_A\|K_2^{\varsigma,1-p}) +$$
$$+ \tau\cdot\partial_{H7}(SS_A^{dp}\|K_2^{\varsigma,p}) =$$

(lemmas 7.12 (ii) and 7.1!)

$$= c2(d,p)\cdot\partial_{H7}(SW_A^{dp}\|K_2^{[d,p],p}) +$$
$$+ s7(d,p)\cdot\partial_{H7}(SS_A\|K_2^{\varsigma,1-p}) +$$
$$+ \tau\cdot\partial_{H7}(SS_A^{dp}\|K_2^{\varsigma,p})$$

$$\partial_{H7}(SW_A^{dp}\|K_2^{\varsigma,1-p}) = \tau\cdot\partial_{H7}(SS_A^{dp}\|K_2^{\varsigma,1-p}) + \quad (6)$$
$$+ r9(fa)\cdot\partial_{H7}(S_A^{1-p}\|K_2^{\varsigma,1-p})$$

$$\partial_{H7}(SW_A^{dp}\|K_2^{\varsigma,p}) = \tau\cdot\partial_{H7}(SS_A^{dp}\|K_2^{\varsigma,p}) + \quad (7)$$

$$\boxed{+ r9(fa)\cdot\partial_{H7}(S_A^{1-p}\|K_2^{\varsigma,p})}$$

$$\partial_{H7}(SS_A\|K_2^{\varsigma,1-p}) = c2(d,p)\cdot\partial_{H7}(SW_A^{dp}\|K_2^{[d,p],1-p}) = \quad (8)$$

(lemmas 7.12(i) and 7.11)

$$= c2(d,p)\cdot\partial_{H7}(SW_A^{dp}\|K_2^{\varsigma,1-p})$$

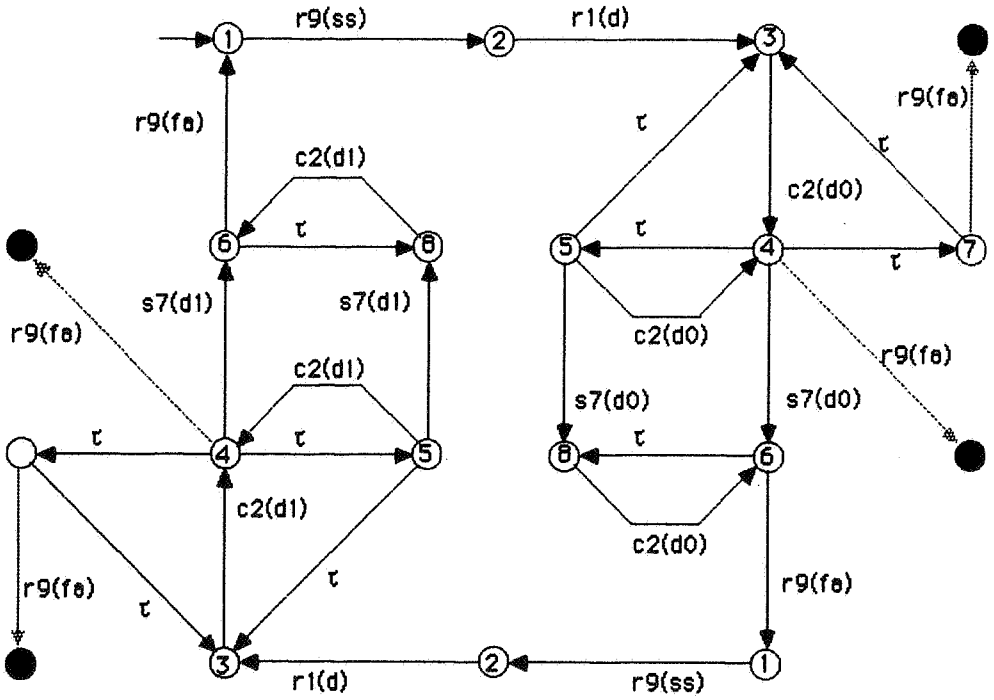


Fig. 30

We claim that the summands in blocks in fact can be deleted. This is because

(i) There exist strictly linear specifications for $\partial_{H7}(S_A^{1-p} \| K_2^{[d,p],p})$ and $\partial_{H7}(S_A^{1-p} \| K_2^{\varsigma,p})$, and therefore also for $\partial_{H7}(S_A \| K_2)$ (theorem 1.5.6).
(ii) In the strictly linear specification for $\partial_{H7}(S_A \| K_2)$, the summands which correspond to the summands in blocks are redundant in context *obsw* by application of lemma 7.14.
(iii) According to theorem 6.16 we can omit the redundant summands in the strictly linear specification of $\partial_{H7}(S_A \| K_2)$.
(iv) By RSP, the solution of the resulting specification is the same as the solution of X.

This finishes the proof of lemma 7.15.

7.16 LEMMA. Let X_{abs} be specified as follows ($p \in B$):

$$\begin{aligned} X_{abs} &= X_{abs}^0 \\ X_{abs}^p &= r9(ss) \cdot \sum_{d \in D} r1(d) \cdot s7(d,p) \cdot r9(fa) \cdot X_{abs}^{1-p} \end{aligned}$$

then

$$OBSW = \tau_{I5} \circ \partial_{H5}(IMP_{A2} \| R_{A3} \| X_{abs} \| IMP_{B2} \| R_{B3} \| S_B \| L_2)$$

PROOF: Let $I_7 = \{c2(f) | f \in DB\}$. Then

$$\begin{aligned} OBSW &= \tau_{I5}(obsw) = \\ &= \tau_{I5} \circ \partial_{H5}(IMP_{A2} \| R_{A3} \| X \| IMP_{B2} \| R_{B3} \| S_B \| L_2) = \\ &= \tau_{I5} \circ \partial_{H5}(IMP_{A2} \| R_{A3} \| \tau_{I7}(X) \| IMP_{B2} \| R_{B3} \| S_B \| L_2) \end{aligned}$$

It is enough to prove

$$X_{abs} = \tau_{I7}(X)$$

Observe that for d and p fixed, X_6^{dp} and x_8^{dp} form a conservative cluster from $I7$. CFAR gives:

$$\tau_{I7}(X_6^{dp}) = \tau_{I7}(X_8^{dp}) = \tau \cdot r9(fa) \cdot \tau_{I7}(X_1^{1-p})$$

For d an p fixed, X_3^{dp}, X_4^{dp}, X_5^{dp} and X_7^{dp} also form a conservative cluster from $I7$. CFAR gives:

$$\begin{aligned} \tau_{I7}(X_3^{dp}) &= \tau \cdot (s7(d,p) \cdot \tau_{I7}(X_6^{dp}) + s7(d,p) \cdot \tau_{I7}(X_8^{dp})) = \\ &= \tau \cdot s7(d,p) \cdot r9(fa) \cdot \tau_{I7}(X_1^{1-p}) \end{aligned}$$

The rest of the proof is straightforward:

$$\begin{aligned} \tau_{I7}(X) &= \tau_{I7}(X_1^0) \\ \tau_{I7}(X_1^p) &= r9(ss) \cdot \sum_{d \in D} r1(d) \cdot s7(d,p) \cdot r9(fa) \cdot \tau_{I7}(X_1^{1-p}) \end{aligned}$$

Now apply RSP.

Analogously we can prove the following lemma:

7.17 LEMMA. Let Y_{abs} be specified as follows ($p \in B$):

$$Y_{abs} = r11(ss) \cdot \sum_{d \in D} r5(d) \cdot r11(fa) \cdot r11(ss) \cdot s3(d, 0) \cdot r11(fa) \cdot Y^1_{abs}$$

$$Y^p_{abs} = r11(ss) \cdot \sum_{d \in D} r5(d) \cdot s3(d,p) \cdot r11(fa) \cdot Y^{1 \leftarrow p}_{abs}$$

then

$$OBSW = \tau_{I5} \circ \partial_{H5}(IMP_{A2} \| R_{A3} \| X_{abs} \| IMP_{B2} \| R_{B3} \| Y_{abs})$$

PROOF:-

Lemma 5.15 up to and including lemma 7.17 were preparations for the following theorem, which shows that the OBSW-protocol is a valid communication protocol (although not very efficient).

7.18 THEOREM.

$$OBSW = \tau \cdot (\sum_{d \in D} r1(d) \cdot \sum_{e \in D} r5(e) + \sum_{e \in D} r5(e) \cdot \sum_{d \in D} r1(d)) \cdot s8(d) \cdot s4(e) \cdot OBSW'$$

$$OBSW' = \sum_{d \in D} r1(d) \cdot s8(d) \cdot \sum_{e \in D} r5(e) \cdot s4(e) \cdot OBSW'$$

PROOF: Straightforward. To handle the nondeterminism at the beginning, use axioms T1 and T2.

REFERENCES

[1] BAETEN, J.C.M. & J.A. BERGSTRA, *Global renaming operators in concrete process algebra,* CWI Report CS-R8521, Amsterdam 1985.

[2] BAETEN, J.C.M. , J.A. BERGSTRA & J.W. KLOP, *Conditional axioms and α / β calculus in process algebra,* CWI Report CS-R 8502, Amsterdam 1985.

[3] BAETEN, J.C.M., J.A. BERGSTRA & J.W. KLOP, *Syntax and defining equations for an interrupt mechanism in process algebra,* CWI Report CS-8503, Amsterdam 1985.

[4] BAETEN, J.C.M., J.A. BERGSTRA & KLOP, *On the consistency of Koomen's fair abstraction rule,* CWI Report CS-8511, Amsterdam 1985.

[5] BAKKER, J.W. DE, *Mathematical Theory of Program Correctness,* Prentice-Hall International series in computer science, London, 1980.

[6a] BERGSTRA, J.A., *Put and Get, primitives for synchronous unreliable message passing,* LGPS No. 3, Department of Philosophy, University of Utrecht, 1985.

[6] BERGSTRA, J.A., & J.W. KLOP, *Algebra of communicating processes with abstraction,* Theor. Comp. Sci. 37(1), pp. 77-121, 1985.

[7] BERGSTRA, J.A., & J.W. KLOP , *Verification of an alternating bit protocol by means of process algebra,* report CS-R8404, Amsterdam 1984.

[8] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes,* CWI Report CS-R8421, to be published in: Proceedings of the CWI Symposium Mathematics and Computer Science (eds. J.W. de Bakker, M. Hazewinkel and J.K Lenstra), North-Holland, Amsterdam 1986.

[9] GLABBEEK, R.J. VAN, personal communication.

[10] HOPCROFT, J.E. & J.D. ULLMAN, *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley, 1979.

[11] REM, M., *Partially ordered computations, with applications to VLSI design,* in: Proc. 4th Advanced Course on Found. of Comp. Sci. part 2, eds. J.W. de Bakker & J. van Leeuwen, MC Tract 159, Mathematical centre, pp. 1-44, Amsterdam 1983.

[12] TANENBAUM, A.S., *Computer Networks,* Prentice Hall, 1981.