A MODULAR APPROACH TO PROTOCOL VERIFICATION USING PROCESS ALGEBRA

C.P.J. Koymans, Department of Philosophy,
State University of Utrecht.

J.C. Mulder, Mathematical Institute,
State University of Utrecht

Logic Group
Preprint Series
No. 6

Department of Philosophy

University of Utrecht

A Modular Approach to Protocol Verification using Process Algebra

C.P.J. Koymans,
Department of Philosophy, State University of Utrecht.

J.C. Mulder,
Mathematical Institute, State University of Utrecht.

A version of the Alternating Bit Protocol is verified by means of Process Algebra. To avoid a combinatorial explosion, a notion of "modules" is introduced and the protocol is divided in two such modules. A method is developed for verifying conglomerates of modules and applied to the motivating example.

1985 Mathematical Subject Classification: 68Q10, 68Q45, 68Q55.
1982 CR Categories: F.1.1, F.1.2, F.3.2, C.2.2.
Key Words & Phrases: process algebra, concurrency, communication protocol, verification, fairness, module.
Note: This research was partially supported by Esprit project Meteor (contract no. 432).

Introduction.

One of the basic problems in protocol verification is the following: data are to be transmitted from A to B via some unreliable medium M. A protocol has been proposed for doing so correctly and perhaps efficiently. A rigorous mathematical proof of the correctness claim is desired.

Now protocol verification aims at providing the techniques for giving such a proof. Several formalisms have been advocated, but as yet none has been widely accepted. For a survey of current research, see [2].

The framework we adhere to is Process Algebra. The first protocol correctness proof by means of Process Algebra is in Bergstra and Klop [4], where a simple version of the Alternating Bit Protocol is verified.

We have tried our hands at a more complicated version, called the Concurrent Alternating Bit Protocol (CABP) and found that the number of possible state transitions was prohibitively large. In this paper we propose a divide-and-conquer strategy. We group processes into modules, describe and verify their behaviour and finally combine them. For different approaches, see [8], [9], [10] and [11].

The structure of this paper then, is as follows:

1. The Concurrent Alternating Bit Protocol
2. The modular approach
3. Modules
4. A verification of CABP
5. References

1. The Concurrent Alternating Bit Protocol.

1.1. Architecture.

The architecture of the protocol can be depicted as in fig 1.1:

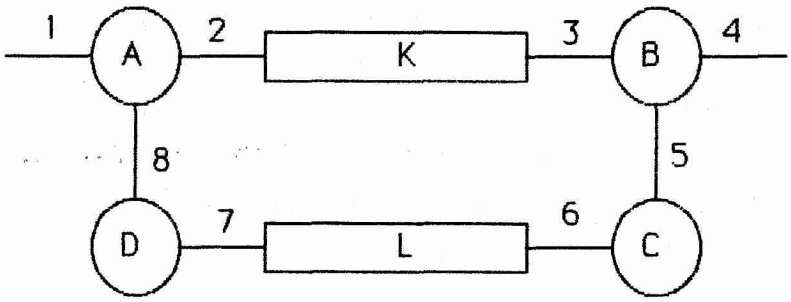


fig 1.1

There are six components:

A: Data transmitter. A reads data from port 1 and transmits them repeatedly via channel K until an acknowledgement has been received from D.

K: Data transmission channel. K transfers data from A to B and may make two sorts of mistakes: a datum may be corrupted, i.e. changed into some error value e recognizable as such, or it may be lost altogether. However K is supposed to be fair in the sense that it will not make infinitely many mistakes consecutively.

B: Data receiver. B receives data from K, outputs them at port 4, and sends an acknowledgement to C via port 5.

C: Acknowledgement transmitter. C receives an acknowledgement from B and repeatedly transmits it via L to D.

L: Acknowledgement transmission channel. L transfers acknowledgements from C to D. It makes similar mistakes as K.

D: Acknowledgement receiver. D receives acknowledgements from L and passes them on to A.

1.2. Remarks.

1.2.1. One might propose to collapse A and D into a sender process, and B plus C into a receiver process. The resulting processes would be more

complicated and in the correctness proof we would have to decompose them again. Consequently, we present them as separate processes in the first place. Of course, if the reader feels more comfortable when he thinks of A and D as running interleaved on the same physical processor, he is free to do so.

1.2.2. This version of the Alternating Bit Protocol is called "concurrent" to emphasize the fact that A and C do not wait for a negative response before retransmitting. The idea is that one can freely retransmit in what otherwise would have been idle time, whereas performance gain will occur if the channel malfunctions, because in this version retransmissions start earlier and occur more often.

1.2.3. In [4] it was assumed that neither K nor L ever "forget" a datum, whereas in [11] a timeout mechanism was added to overcome such a mishap. In Petri-net terms, those protocols pass around a single token. If it ever gets lost, things stop moving, unless a time-out mechanism introduces a fresh token.

In the CABP protocol however, processes A and C keep on firing, so that the rest even has to throw away many tokens to prevent the system from being flooded. In any case, activity never stops, for C never waits for input.

1.3. Data and actions.

1.3.1. Data.

D is the finite set of data to be transferred from port 1 to port 4.
$B = \{0,1\}$ is the set of acknowledgement bits sent at port 6.
$D \times B$, the cartesian product set, will be transferred at port 2.
$B \cup \{e\}$, where e is the error value, is used at port 7.
$D \times B \cup \{e\}$ may be sent at port 3.
$\{ac\}$, where ac is an acknowledgement occurs at ports 5 and 8.
$\mathbf{D} = D \cup D \times B \cup \{0,1,e,ac\}$ is the set of all transferable data.

1.3.2. Actions.

For $d \in \mathbf{D}$ and $p \in \{1,\ldots,8\}$ there are read, send and transfer actions:
$r(d,p)$: read datum d at port p.
$s(d,p)$: send datum d at port p.
$t(d,p)$: transfer datum d at port p.
In fact $t(d,p)$ is a communication action: $t(d,p) = r(d,p) \mid s(d,p)$.
There are six more atomic actions. They are called ik, jk, kk, il, jl and kl and they are internal actions of K and L corresponding to internal choices.

And, of course, there are the inevitable constants δ and τ.

The entire alphabet is then:

$$A = \{r(d,p), s(d,p), t(d,p) \mid d \in \mathbf{D}, 1 \le p \le 8\} \cup \{ik, jk, kk, il, jl, kl\}$$

The communication function $\cdot|\cdot : A \times A \to A$ is:

$a|b = c$ if $\exists d \in \mathbf{D}, p \in \{1,\ldots,8\}$ s.t. $\{a,b\} = \{r(d,p), s(d,p)\} \wedge c = t(d,p)$

$= \delta$ otherwise.

Two relevant subsets of A are:

$H = \{r(d,p), s(d,p) \mid d \in \mathbf{D}, p \in \{2,3,5,6,7,8\}\}$

$I = \{t(d,p) \mid d \in \mathbf{D}, 1 \le p \le 8\} \cup \{ik, jk, kk, il, jl, kl\}$

The process we are investigating is $\tau_I \partial_H(A \| K \| B \| C \| L \| D)$, where A,....,D are specified in 1.4.

1.4. The individual components.

We will describe each component twice, viz. by a state transition diagram and by a recursive set of equations:

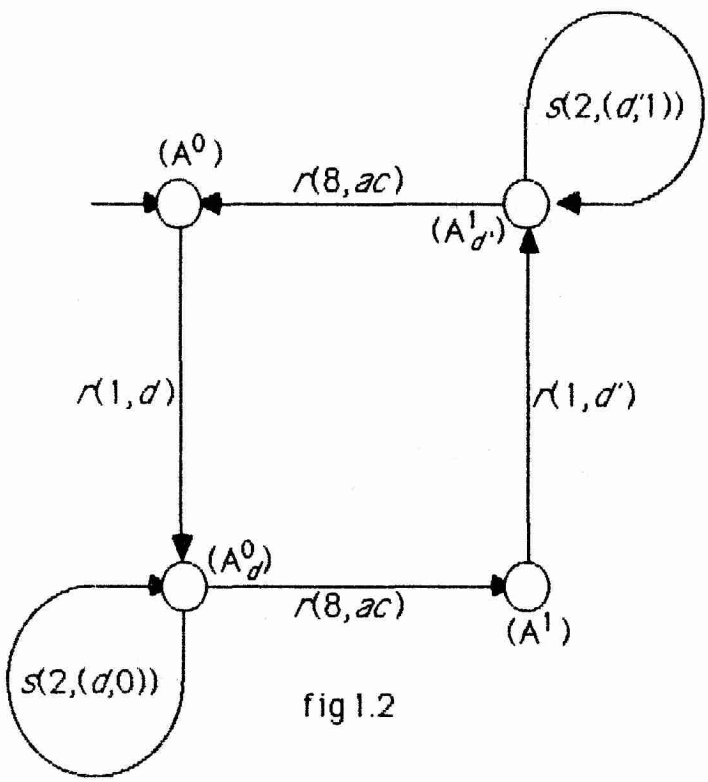


fig 1.2

A:

$$A = A^0$$

$$A^b = \sum_{d \in D} r(1,d) \cdot A^b_d$$

$$A^b_d = s(2,(d,b)) \cdot A^b_d + r(8,ac) \cdot A^{1-b}$$

K:

$$K = \sum_{x \in D \times B} r(2,x) \cdot (ik \cdot s(3,x) + jk \cdot s(3,e) + kk) \cdot K$$

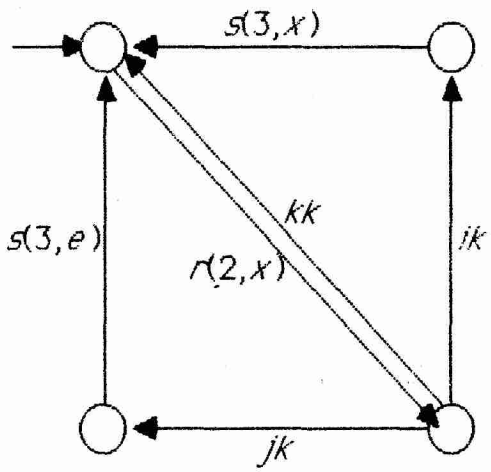


fig 1.3

B:

$$B = B^0$$

$$B^b = (r(3,e) + \sum_{d \in D} r(3,(d,1-b))) \cdot B^b + \sum_{d \in D} r(3,(d,b)) \cdot s(4,d') \cdot s(5,ac) \cdot B^{1-b}$$

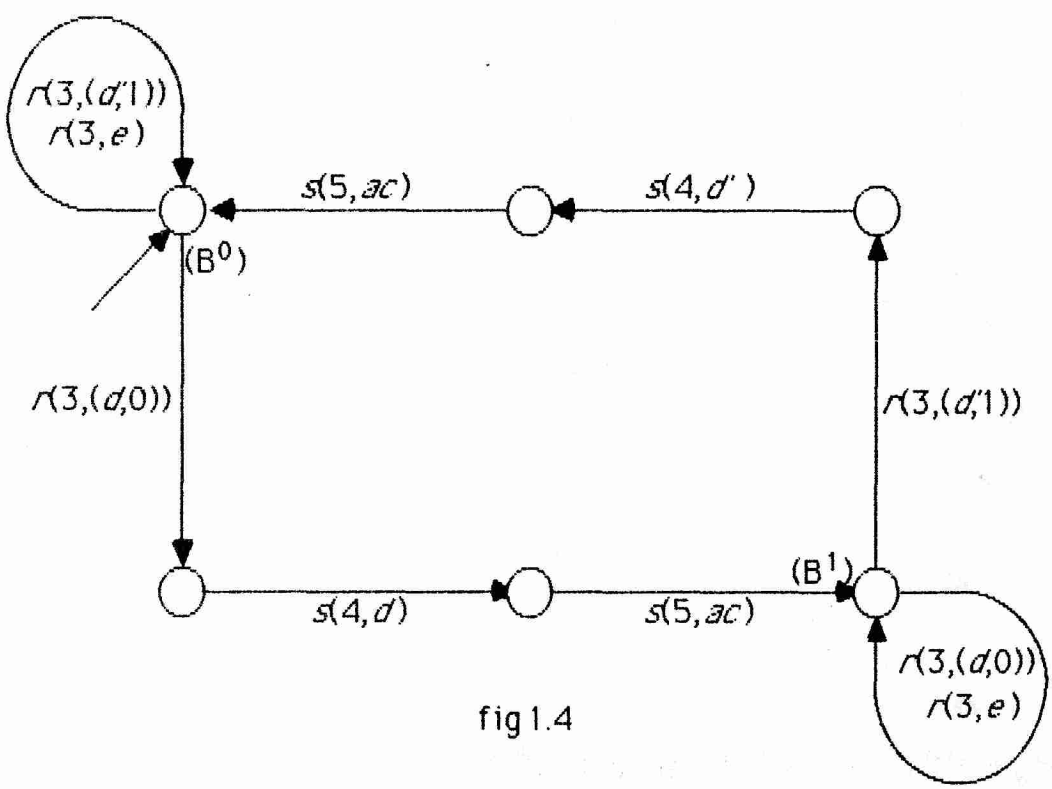
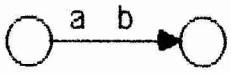
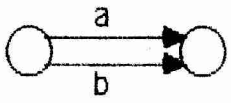


fig 1.4

Note: a doubly labeled arrow 

stands for two arrows 

C:

$C = C^1$ (*not* C^0)

$C^b = r(5, ac) \cdot C^{1-b} + s(6, b) \cdot C^b$

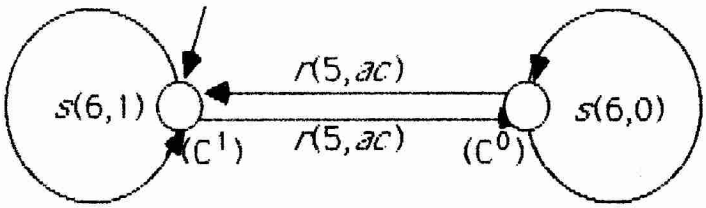


fig 1.5

L:

$$L = \sum_{b \in B} r(6, b) \cdot (il \cdot s(7, b) + jl \cdot s(7, e) + kl) \cdot L$$

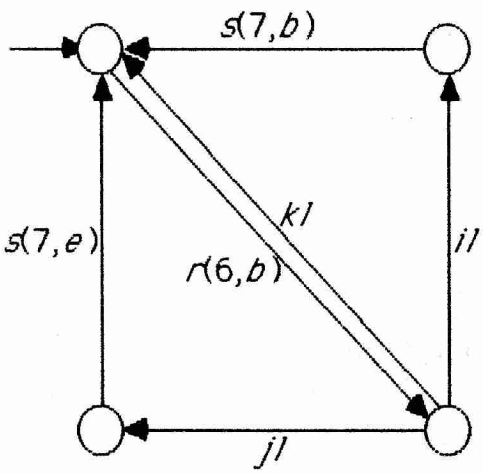


fig 1.6

D:

$$D = D^0$$

$$D^b = (r(7,e)+r(7,1-b)) \cdot D^b + r(7,b) \cdot s(8,ac) \cdot D^{1-b}$$

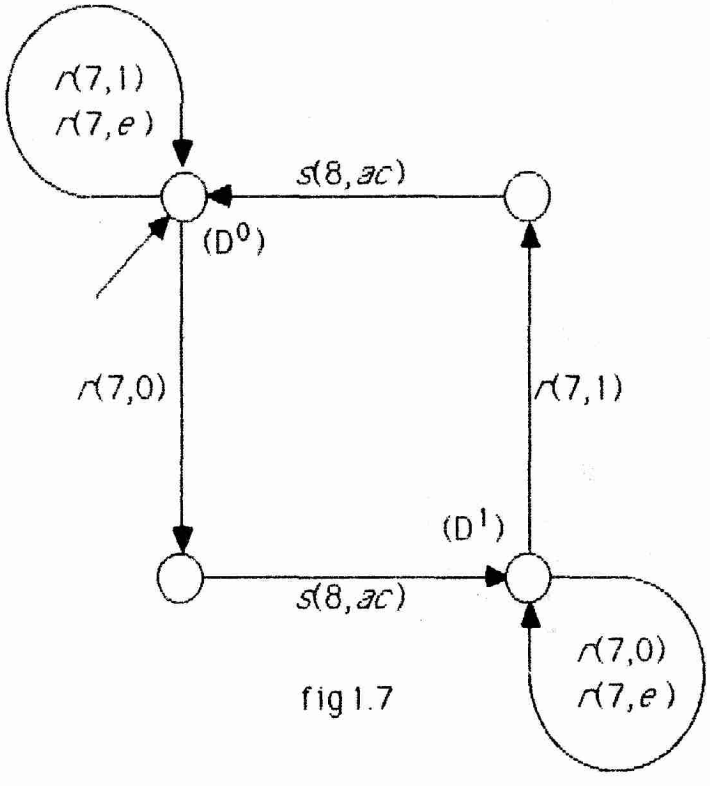


fig 1.7

1.5. Process Algebra.

1.5.1. The reader is supposed to be familiar with Process Algebra, as described in e.g. [3]. For definiteness we list here the features used in the sequel. The most concise description of this combination is

ACP_τ+RDP+RSP+CFAR+HA+SC+CA.

We will briefly comment on each of these features.

1.5.2. ACP_τ, the Algebra of Communicating Processes, with silent step.

This is the kernel system. Its signature is:

a	atomic action ($a \in A$)
$+$	alternative composition (sum)
$\cdot$	sequential composition (product)
$\|\|$	parallel composition (merge)
$\lfloor\!\lfloor$	left-merge
$\|$	communication merge
∂_H	encapsulation ($H \subseteq A$)
τ_I	abstraction ($I \subseteq A$)
δ	deadlock/failure
τ	silent action

Table 1.

The axioms of ACP_τ are in Table 2:

$x + y = y + x$	A1		
$x + (y + z) = (x + y) + z$	A2	$x\tau = x$	T1
$x + x = x$	A3	$\tau x + x = \tau x$	T2
$(x + y)\cdot z = xz + yz$	A4	$a(\tau x + y) = a(\tau x + y) + ax$	T3
$(xy)z = x(yz)$	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		
$x \lVert y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x\|y$	CM1		
$a \lfloor\!\lfloor x = ax$	CM2	$\tau \lfloor\!\lfloor x = \tau x$	TM1
$(ax) \lfloor\!\lfloor y = a(x \lVert y)$	CM3	$(\tau x) \lfloor\!\lfloor y = \tau(x \lVert y)$	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 2.

ACP_τ is a complete axiomatisation of the identities between *finite* processes in the intended model (process graphs modulo rooted τ-bisimulation). However, several desirable principles are not derivable from ACP_τ, even though all of their finite instances are. Being true in the intended model, they can be consistently added. In the next five subsections we briefly explain those principles used in this paper. Some of them have been simplified to facilitate the discussion as we will not use the more general case.

1.5.3. RDP + RSP, Recursice Definition Principle and Recursive Specification Principle.

Suppose $X = \{X_i \mid i \in I\}$ is a set of variables, where I is some index set. Then a set $E = \{E_i \mid i \in I\}$ of equations is called a *specification* of X if each E_i is of the form $X_i = T_i(X)$, where $T_i(X)$ is a process term over X.

A collection $P = \{P_i \mid i \in I\}$ of processes is called a *solution* of E if it satisfies all equations $P_i = T_i(P)$. Notation: $P \models E$.

Usually one of the variables, say X_0, is highlighted to indicate that E was devised to describe the behaviour of P_0. As we feel that this highlighting practice does not clarify matters, we will ignore it.

It may be that the terms T_i in the right hand sides of E contain variables from some set Y disjoint from X. The elements of Y are then called *parameters* of E. As before, if two sets P and Q of processes satisfy the equations $P_i = T_i(X:=P, Y:=Q)$, then P is called a solution for parameter values Q, notation $P \vdash T(Q)$.

Specifications are mostly intended to describe their unique solutions. Unfortunately, a specification as described above does not necessarily have a unique solution. For example, $X=X+a$ does not. A notion called *guardedness* has been developed, such that each *guarded* specification does have a *unique* solution. The precise definition of guardedness is rather complicated and in this paper we can manage with a special case:

(*) If the terms T_i in the RHS of a specification E are all of the form $\sum_j a_{ij}T_{ij}+U_i$ with $a_{ij} \in A\setminus\{\tau\}$, T_{ij} a term built from $A \cup X \cup Y$ using only + and ·, and U_i a term built from $A \cup Y$, then E is guarded.

Readers not familiar with guardedness should either read [6], or regard (*) as a definition.

Now RDP says that guarded specifications do have solutions, and RSP says that they are unique:

Let E be a guarded specification, then:	
(RDP) $\exists X: X \vdash E(Y)$	(RSP) $\dfrac{X \vdash E(Y) \qquad X' \vdash E(Y)}{X = X'}$

1.5.4. CFAR, the Cluster Fair Abstraction Rule.

According to the latest insights, this, or at least the version without the parameters Y_i, is just a consequence of Koomen's Fair Abstraction Rule, see [11].

Consider a specification E of the form $\{X_i=\sum_j a_{ij}X_j + Y_i \mid i \in I\}$. Define a relation R on $X \times X$ by putting $X_i R X_j$ iff a term $a_{ij}X_j$ with $a_{ij} \neq \delta$ occurs in T_i above. E is called a *cluster* if the transitive closure $\mathbf{R}$ of R is the trivial relation $X \times X$, i.e. $\forall ij: X_i \mathbf{R} X_j$.

Pictorially a cluster is a set of nodes in which each node can be reached from anywhere in the set.

CFAR expresses a fairness assumption: if a process is in a state within a cluster and there is a way out, the process will eventually get out.

Stated more formally, if we abstract from the internal transitions $X_i \xrightarrow{a_{ij}} X_j$, we will see a silent move τ, followed by one of the Y_i:

If E is a cluster as specified above, if $I \subseteq A$, if all of the a_{ij} above are elements of I, then:

$$\text{(CFAR)} \quad \frac{X \models E(Y)}{\tau_I(X_i) = \tau \cdot \tau_I(\Sigma Y)}$$

1.5.5.1.HA, the Handshaking Axiom.
This one is less finicky: Communication Is Binary.

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

1.5.5.2.SC, Standard Concurrency.
The first impression one gets from these axioms is that they are easliy derived from CM1-9 by induction. For finite terms this is true. For infinte processes however, the induction lacks a basis. Hence these equations are introduced as extra axioms about the merge operators.

$(x \mathbin{\lfloor\!\lfloor} y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} (y \| z)$	SC1
$(x \mid ay) \mathbin{\lfloor\!\lfloor} z = x \mid (ay \mathbin{\lfloor\!\lfloor} z)$	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 3.

1.5.5.3.ET, the Expansion Theorem.
Like the name suggests, ET is not an axiom but a theorem. It was first proven by Milner in [7] in the context of CCS. In [3] it occurs as a theorem of ACP_τ+SC+HA.

ET is intuitively obvious, but not easily formulated. It describes the result of working out the parallel merge of a lot of processes, assuming HA+SC. In fact, it is meaningless without at least SC4+6.

To formulate ET we need a little extra notation: if $T=\{T_1,\ldots,T_n\}$ is a finite multiset of terms,then $\|T = T_1\|T_2\|\ldots\|T_n$.

If X is finite, then:

$$(ET)\quad \|X = \sum_{x\in X} x \between (\|(X-\{x\})) + \sum_{\substack{x,y\in X\\ x\neq y}} (x|y) \between (\|(X-\{x,y\}))$$

1.5.6. CA, the Conditional Axioms.
CA is a set of seven rules, introduced in [1]. The name refers to the fact that the rules contain conditions on the alphabets above the line.

The alphabet $\alpha(T)$ of a term T is just the set of atomic actions in it. To be more precise, it is the set of atomic actions T can potentially perform. For example $\alpha(a\delta b + \tau_{\{c\}}(cd)) = \{a,d\}$.

α is defined by:

$$\begin{array}{l}\alpha(\delta) = \emptyset\\ \alpha(\tau) = \emptyset\\ \alpha(a) = \{a\}\\ \alpha(x+y) = \alpha(x) \cup \alpha(y)\\ \alpha(ax) = \{a\} \cup \alpha(x)\end{array}$$

Table 4.

The CA rules themselves are in table 5:

$$\frac{\alpha(x)|(\alpha(y)\cap H)\subseteq H}{\partial_H(x\|y)=\partial_H(x\|\partial_H(y))}\ (CA1) \qquad \frac{\alpha(x)|(\alpha(y)\cap I)=\emptyset}{\tau_I(x\|y)=\tau_I(x\|(\tau_I y))}\ (CA2)$$

$$\frac{\alpha(x)\cap H = \emptyset}{\partial_H(x) = x}\ (CA3) \qquad \frac{\alpha(x)\cap I = \emptyset}{\tau_I(x) = x}\ (CA4)$$

$$\frac{H = H_1 \cup H_2}{\partial_H(x)=\partial_{H1}\circ\partial_{H2}(x)}\ (CA5) \qquad \frac{I = I_1 \cup I_2}{\tau_I(x)=\tau_{I1}\circ\tau_{I2}(x)}\ (CA6)$$

$$\frac{H\cap I = \emptyset}{\tau_I\circ\partial_H(x)=\partial_H\circ\tau_I(x)}\ (CA7)$$

Table 5.

2. The Modular Approach.

2.1. Running example.

When irrelevant details are stripped off, CABP consists of six processes looking like this:

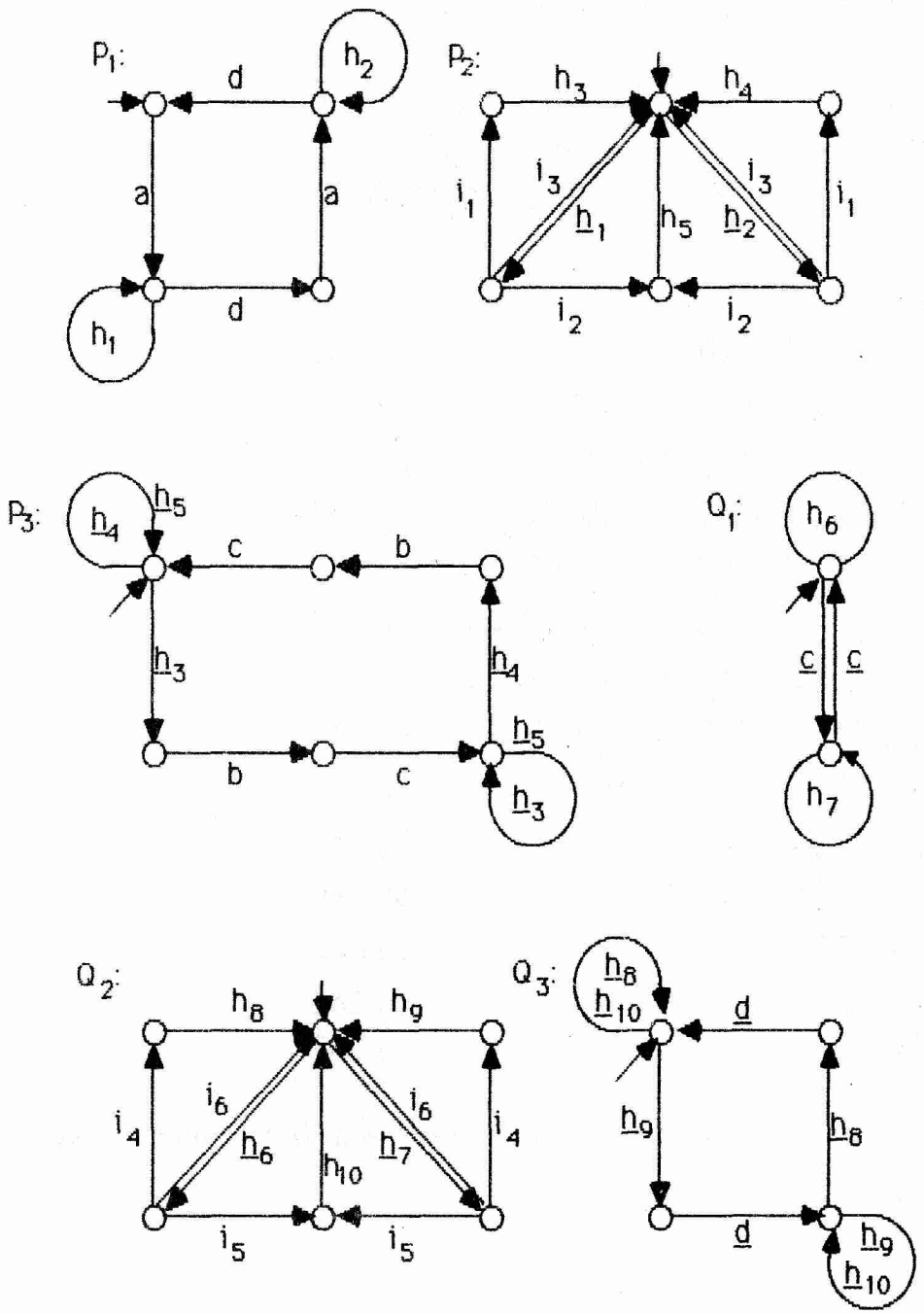


fig 2.1

Here $\ell | \underline{\ell} = \ell^\circ$ for those ℓ that occur underlined.

We put P_1, P_2, P_3, Q_1, Q_2, Q_3 for A, K, B, C, L, D, respectively. The atomic actions have been given more systematic names, too.

Now put $H = \{h_1, \underline{h}_1, \ldots, \underline{h}_{10}\}$. $H^+ = H \cup \{c, \underline{c}, d, \underline{d}\}$, $I = \{h_1^\circ, \ldots, h_{10}^\circ, i_1, \ldots, i_6\}$, and $I^+ = I \cup \{c^\circ, d^\circ\}$.

CABP being a communication protocol, $\tau_I\partial_H(P_1 \| P_2 \| P_3 \| Q_1 \| Q_2 \| Q_3)$ should equal $(\tau\cdot)(ab)^\omega$, where $(\tau\cdot)x$ stands for "either x or $\tau\cdot x$". Due to a combinatorial explosion, this is not easy to prove.

2.2. The CA axioms allow a significant reduction:

Put $H_P = \{h_1, \underline{h}_1, \ldots, \underline{h}_5\}$. $I_P = \{h_1^\circ, \ldots, h_5^\circ, i_1, i_2, i_3\}$,
$H_Q = \{h_6, \underline{h}_6, \ldots, \underline{h}_{10}\}$. $I_Q = \{h_6^\circ, \ldots, h_{10}^\circ, i_4, i_5, i_6\}$,
$P = \tau_{I_P}\partial_{H_P}(P_1 \| P_2 \| P_3)$, $Q = \tau_{I_Q}\partial_{H_Q}(Q_1 \| Q_2 \| Q_3)$

Then
$\tau_I\partial_H(P_1 \| P_2 \| P_3 \| Q_1 \| Q_2 \| Q_3)$
$= \tau_I\partial_H\partial_{H_P}(P_1 \| P_2 \| P_3 \| Q_1 \| Q_2 \| Q_3)$ by CA5
$= \tau_I\partial_H\partial_{H_P}(\partial_{H_P}(P_1 \| P_2 \| P_3) \| Q_1 \| Q_2 \| Q_3)$ by CA1
$= \tau_I\partial_H(\partial_{H_P}(P_1 \| P_2 \| P_3) \| Q_1 \| Q_2 \| Q_3)$ by CA5
$= \tau_I\partial_H\partial_{H_Q}(\partial_{H_P}(P_1 \| P_2 \| P_3) \| Q_1 \| Q_2 \| Q_3)$ by CA5
$= \tau_I\partial_H\partial_{H_Q}(\partial_{H_P}(P_1 \| P_2 \| P_3) \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA1
$= \tau_I\partial_H(\partial_{H_P}(P_1 \| P_2 \| P_3) \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA5
$= \partial_H\tau_I(\partial_{H_P}(P_1 \| P_2 \| P_3) \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA7
$= \partial_H\tau_I\tau_{I_P}(\partial_{H_P}(P_1 \| P_2 \| P_3) \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA6
$= \partial_H\tau_I\tau_{I_P}(P \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA2
$= \partial_H\tau_I(P \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA6
$= \partial_H\tau_I\tau_{I_Q}(P \| \partial_{H_Q}(Q_1 \| Q_2 \| Q_3))$ by CA6
$= \partial_H\tau_I\tau_{I_Q}(P \| Q)$ by CA2
$= \partial_H\tau_I(P \| Q)$ by CA6
$= \tau_I\partial_H(P \| Q)$ by CA7

So we might first calculate P and Q separately, and then combine them.

This would yield:

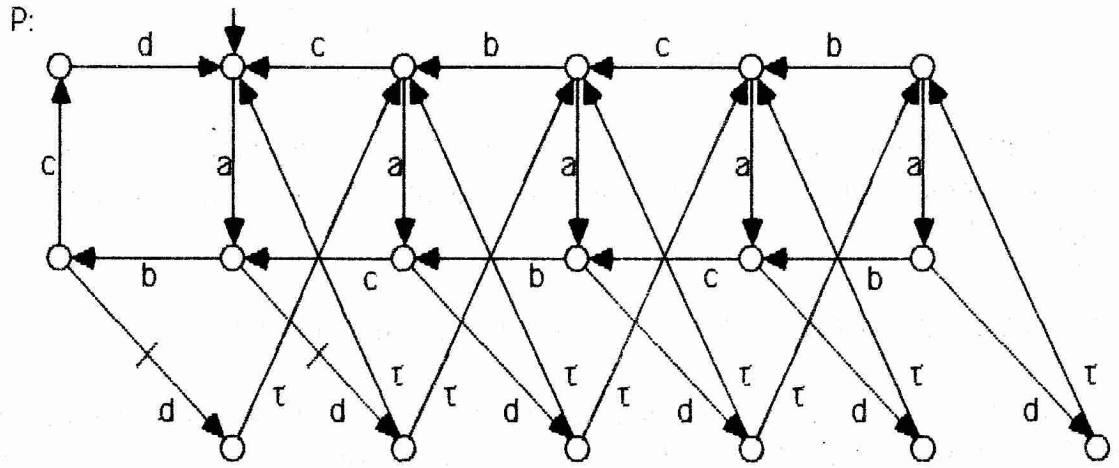


fig 2.2

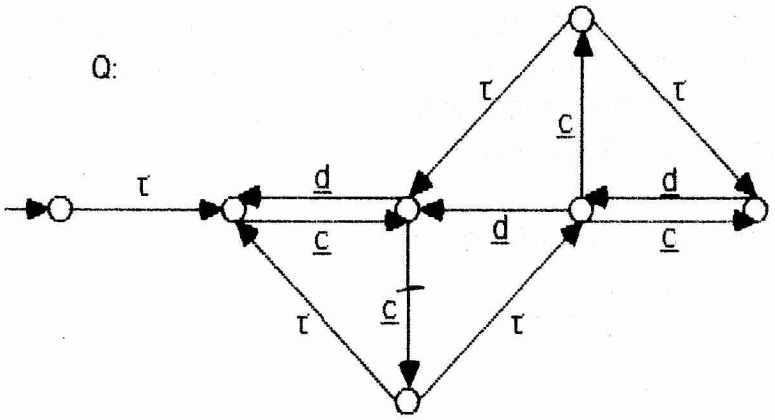


fig 2.3

When merging these processes, one finds that the edges marked with a crossbar (⟶+⟶) are redundant in the sense of Vaandrager [11], i.e. they do not communicate and are encapsulated out of the picture.

It is evident that any edge that can be reached from the root only via one of these redundant edges is itself redundant, too. Inspection of fig 2.2 and 2.3 shows that this is the case in about 80% of both P and Q. So 80% of these graphs is irrelevant information. We only need to know that they look somewhat like this:

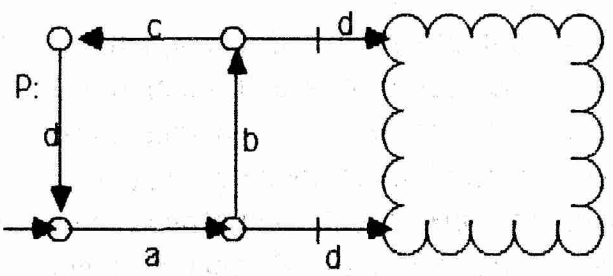


fig 2.4

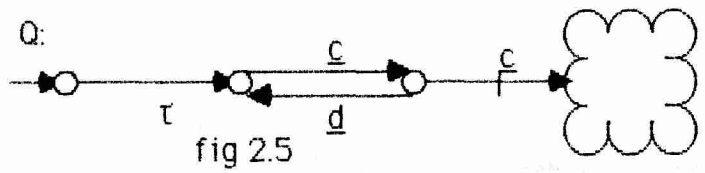


fig 2.5

Figure 2.4 and 2.5 are evidently much clearer than figure 2.2 and 2.3 and yet they contain all relevant information for showing $\tau_I\partial_H(P\|Q) = \tau\cdot(abc^\circ d^\circ)^\omega$.

The rest of this paper is mainly devoted to formalizing these ideas.

2.3. In 2.2 a process graph P was constructed and then it was argued that 80% of it was redundant and could be thrown away. Hence 80% of the calculations needed to find P were redundant and should have been avoided. That is, instead of laboriously deriving fig 2.2 and 2.3 from the CABP specification and then simplifying to fig 2.4 we should have conjectured fig 2.4 from the informal description in section 1, and we should have proved this conjecture by methods to be developed in section 3. In fact this program will be carried out in section 4.

2.4. Figure 2.6 is a simplified version of fig 1.1.

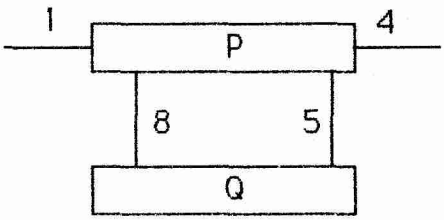


fig 2.6

Recall that P is supposed to communicate at ports 1, 4, 5 and 8, in that order, repeatedly. In this section we denote these communications as a, b, c and d, so a first approximation to P should be $P^* = abcd\cdot P^*$.

2.5. In 2.4 it was argued that P "should be" $(abcd)^\omega$. Of course, it isn't. For, if it were, process Q would not be needed. Q is to pass signals from port 5 to port 8. As these signals do not contain any information, all Q actually does, is to see to it that communication at port 8 occurs later than communication at port 5.

P is ready to communicate at port 8 at any time after communicating at 1. In contrast, the other ports of P only communicate when it is their turn. We will call a port *robust* if it always awaits its turn before communicating. Here "its turn" is defined by a specification, so in case of doubt we may have to talk about robustness relative to a certain specification. For example, in process P, port 8 is not robust relative to

P^*, port 5 is not robust relative to abδ and all ports are robust relative to P itself.

As a general design principle, whenever we connect two modules via a channel, we will see to it that at least one end is a robust port and hence blocks the channel when it should be inactive. Usually the robust end will be thought of as sending the signal.

2.6. Definition. A *module specification* S is a pair (F(S), (R(S)), where F(S) is a process called the *functional part*, and R(S) is a subset of $\alpha(R(S))$ called the *robustness set*.
Think of a module specification as describing an imperfect black box: F(S) describes the intended behaviour, and R(S) describes possible deviations from F(S).

Examples: $S_P = ((abcd)^\omega, \{a,b,c\})$; $S_Q = ((\underline{c}\ \underline{d})^\omega, \{\underline{d}\})$.
Processes P and Q are to "implement" S_P and S_Q respectively.

2.7. In section 3.8. we will define a notion of "P implements M", define an algebraic criterion for this relation, describe how modules are assembled into larger modules, and prove that a robust module meets its specification.

2.8. Remark. The terms "specification" and implementation" are meant to suggest that module implementations are interchangeable: if some assembly of modules performs a certain function, then any implementations of these modules will do the job.
For example, channels K and L in 1.4 can buffer at most one item at a time. It they are replaced by channels with a larger capacity, P and Q still implement S_P and S_Q, so the combination still works.

3. Modules.

3.1. Definition. A *module* is a finitely branching, rooted, directed, connected multigraph with two sorts of edges: *normal* ($\longrightarrow$), and *barred* ($\longrightarrow\!\!\!\!\!\!+\longrightarrow$). Moreover, both sorts of edges are labeled with labels from a finite set $A \cup \{\delta,\tau\}$.

3.2. Notation. **M** is the set of modules. If $M \in \mathbf{M}$, then

r(M) is the root of M,

N(M) is the set of nodes of M,

E(M) is its set of edges,

$\alpha(M)$ is its alphabet, i.e. its set of node labels,

e: $n \xrightarrow{a} m$ means that e is an edge from n to m labeled a,

$\pi: n \xrightarrow{\sigma} m$ means that π is a path from n to m, and the labels along π other than τ spell the word $\sigma \in A^*$. If n=m, π may be empty. Notice that σ never contains δ.

G = $\{M \in \mathbf{M} \mid$ M contains no barred edges$\}$. Elements of **G** are called *robust modules*.

$\leftrightarrows_\tau$ is the usual notion of τ-bisimulation on **G**.

$\leftrightarrows_{r\tau}$ denotes rooted τ-bisimulation on **G**.

3.3. Definition. With each module specification S we associate a module M(S) in the following way:
Take a process graph G of the functional part F(S) of S, with normal edges, and add barred edges $n \xrightarrow{a}\!\!\!\!\!\!+\longrightarrow \cdot$ for each non-robust action a (i.e. $a \in A\text{-}R(S)$) and for each node n from which no edge $n \xrightarrow{a} \cdot$ emerges. The destination of the barred edges is immaterial.

Example: $M(S_p) = M(\ ((abcd)^\omega, \{a,b,c\})\ )$ becomes:

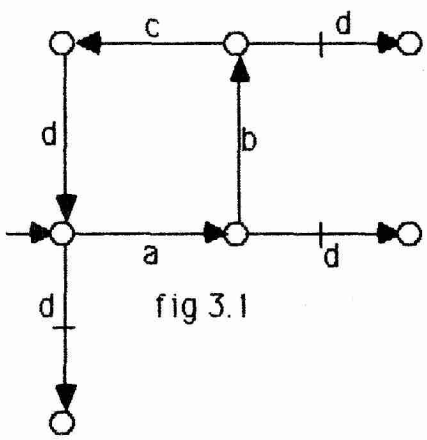


fig 3.1

3.4. The reader might notice that fig 3.1 contains one more $\xrightarrow{d}$ edge than does fig 2.4. This is because fig 3.1 expresses "P might always do d", whereas fig 2.4 is based on a careful analysis of exactly when "P really can do d". Apparently modules can deal with partial robustness, whereas robustness sets cannot. This is why we study graphs rather than algebraic specifications.

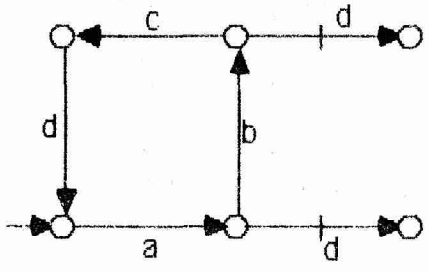


fig 3.2

In the sequel, our running example will be M_p, depicted in fig 3.2. This is the exact counterpart of fig 2.4.

3.5. Definition. A node ℓ is called *loopy* if its outgoing edges are (precisely) $\ell \xrightarrow{a} \ell$ for all $a \in A$, i.e. loops labeled with all letters of the alphabet A

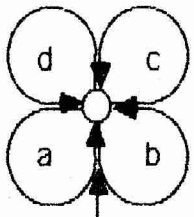
Example: If $A = \{a,b,c,d\}$, then  is loopy.

fig 3.3.

In the sequel we will abbreviate multiple edges connecting the same nodes, and in particular loops, by multiply labeled edges, like this:

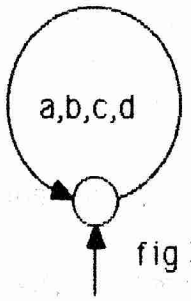


fig 3.4

3.6. Definition. We will need two operations from **M** to **G**:

$|\cdot|$: leave out all barred edges, and

$!\cdot!$: add a loopy node ℓ, and replace each $n \xrightarrow{a} \cdot$ by $n \xrightarrow{a} \ell$.

In these and all subsequent operations on graphs it is tacitly implied that whenever a graph becomes disconnected, only those parts that can be reached from the root are retained.

Examples:

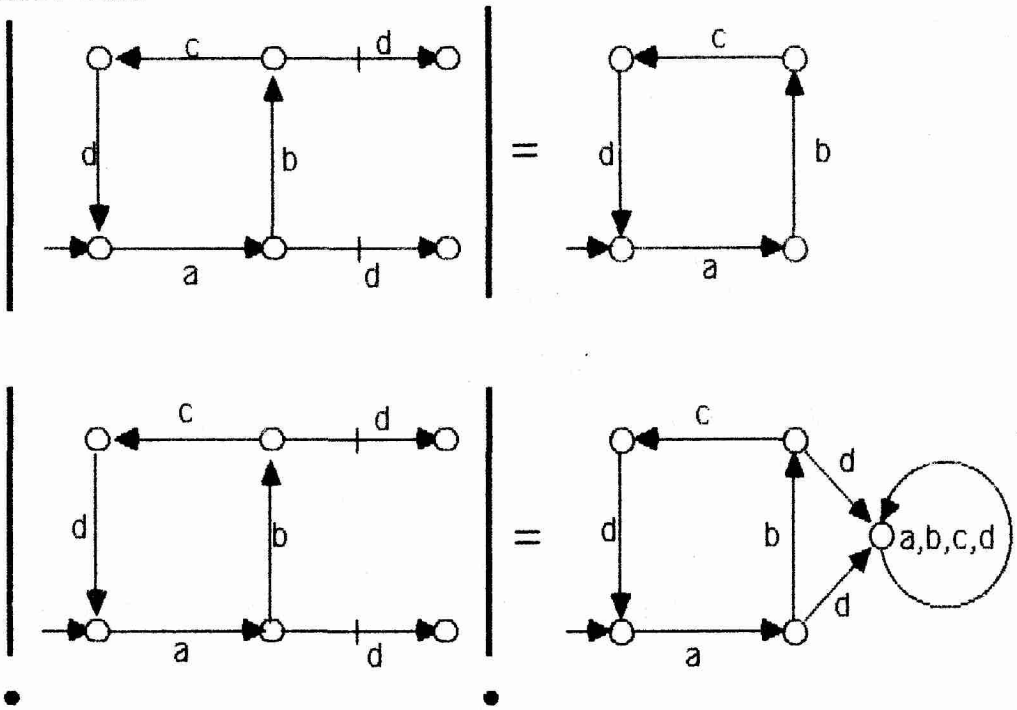


fig 3.5

3.7.1. Definition. Let g,h∈**G**. Then $g \leftarrow h$ (pronounced "g *emulates* h") iff whenever a path π: $r(h) \xrightarrow{\sigma} n$ can be lengthened to $r(h) \xrightarrow{\sigma} n \xrightarrow{\sigma'} m$, then all paths $r(g) \xrightarrow{\sigma} k$ can be lengthened to $r(g) \xrightarrow{\sigma} k \xrightarrow{\sigma'} \ell$.

3.7.2. Remark. Note that if $g \leftarrow h$ and there exists a path $r(h) \xrightarrow{\sigma} n$, then there is at least one path $r(g) \xrightarrow{\sigma} k$, because the empty path $r(g) \xrightarrow{\varepsilon} r(g)$ can be lengthened to $r(g) \xrightarrow{\varepsilon} r(g) \xrightarrow{\sigma} k$, so the condition "for all paths ..." never holds vacuously.

3.7.3. A more playful way to describe emulation is the following two-player game:

Player H runs down a path in graph h. When he traverses a τ-edge, he remains silent, but when he traverses an a-edge, he says "a", and his turn is over. Player G runs down a path in graph g. When H announces he traversed an a-edge, G tries to traverse zero or more τ-edges, and then an a-edge. If he fails, he looses; if he succeeds, it is H's turn again.

Somre obvious rules: both players start at the roots of their own graphs; they do not traverse δ-edges; they do not reveal which path they

are following, but only the labels they encounter; G wins if he does not loose, etc.

Now g emulates h iff G cannot loose, no matter how stupid he may play.

3.7.4. Examples:

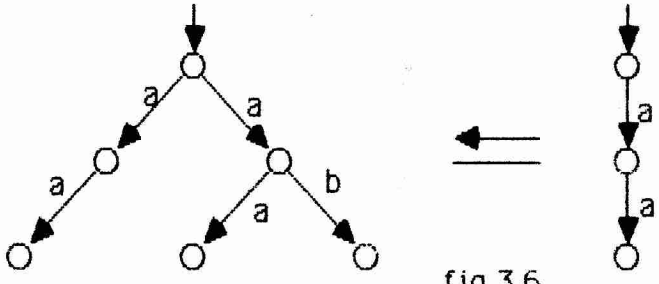


fig 3.6

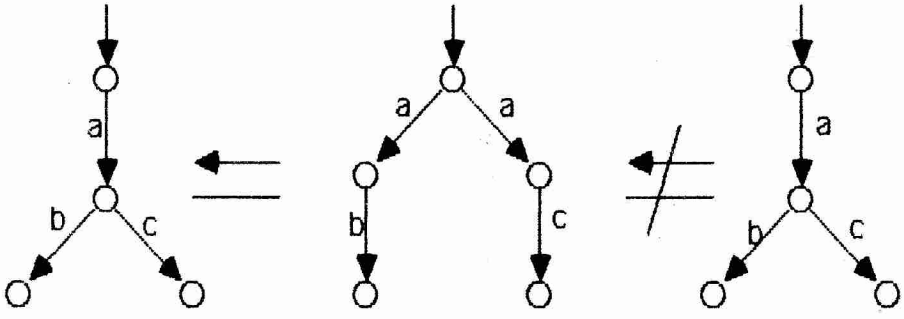


fig 3.7

Emulation is not reflexive, i.o.w. a graph does not necessarily emulate itself:

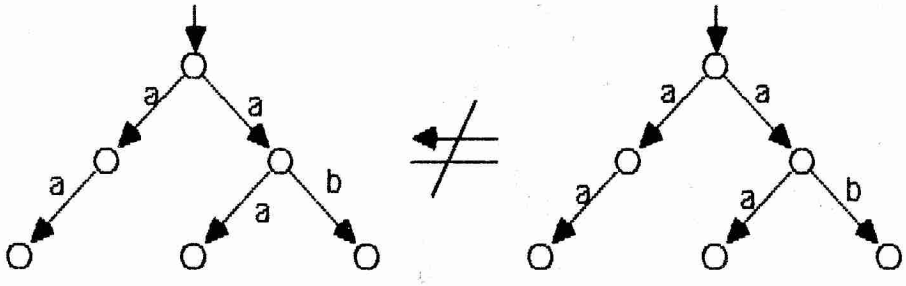


fig 3.8

The rightmost $\xrightarrow{\langle a \rangle}$ can be lengthened to $\xrightarrow{\langle a,b \rangle}$ whereas the leftmost $\xrightarrow{\langle a \rangle}$ can be lengthened to $\xrightarrow{\langle a,a \rangle}$ only.

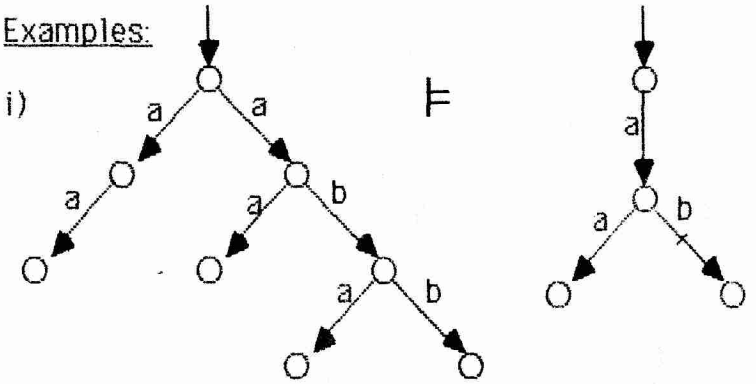
3.7.5. Remark. There are several notions of *emulation*, *simulation* and *implemetation* in the litterature. We invented this one while trying to verify CABP and it turns out that our notion is usable in this case. Experience will show which notion is "best", most universal, easiest to comprehend, etc.

3.8. Definition. The main relation on **M**x**M** is called *implementation*:

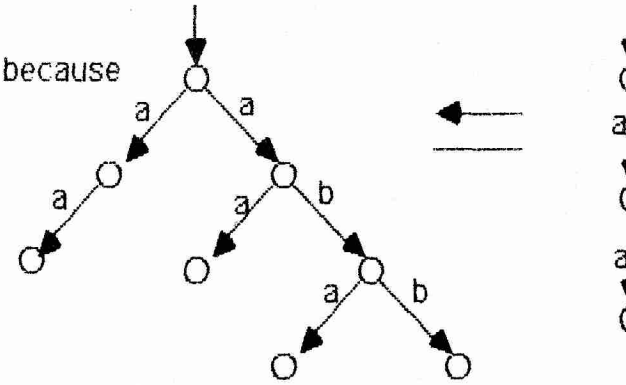
$M \sqsubseteq N \;:\Leftrightarrow\; |M| \leftarrow |N| \;\&\; !N! \leftarrow !M!$

Examples:

i)



because



and

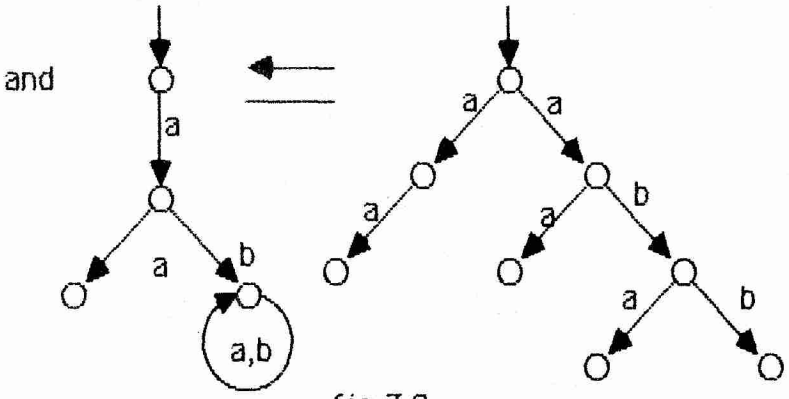


fig 3.9

ii)

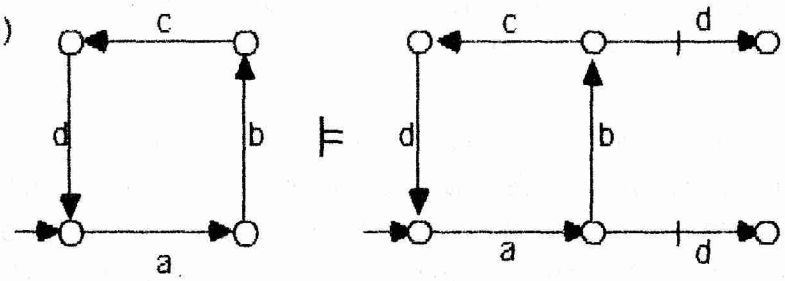


fig 3.10

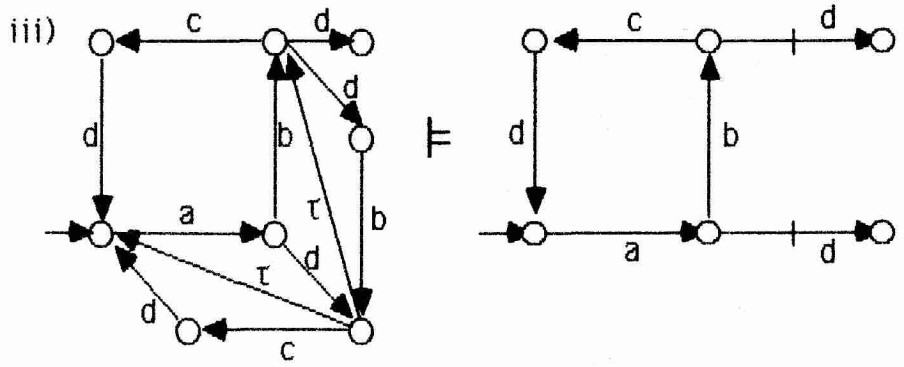


fig3.11

3.9. Lemma: $\cdot \underline{\leftarrow}$ and $\models$ are transitive.

Proof: $\underline{\leftarrow}$: Suppose $g_1 \underline{\leftarrow} g_2$ and $g_2 \underline{\leftarrow} g_3$ and π: $r(g_3) \xrightarrow{\sigma} n_3 \xrightarrow{\sigma'} k_3$ and $r(g_1) \xrightarrow{\sigma} n_1$. Then remark 3.7.2. grants a path $r(g_2) \xrightarrow{\sigma} n_2$. Then there is a path $r(g_2) \xrightarrow{\sigma} n_2 \xrightarrow{\sigma'} k_2$ by the definition of $\underline{\leftarrow}$, and hence $r(g_1) \xrightarrow{\sigma} n_1 \xrightarrow{\sigma'} k_1$.

$\models$: If $M_1 \models M_2$ and $M_2 \models M_3$ then $|M_1| \underline{\leftarrow} |M_2| \underline{\leftarrow} |M_3|$ and $!M_3! \underline{\leftarrow} !M_2! \underline{\leftarrow} !M_1!$ So $|M_1| \underline{\leftarrow} |M_3|$ and $!M_3! \underline{\leftarrow} !M_1!$, hence $M_1 \models M_3$. □

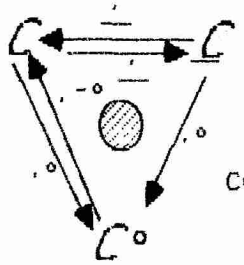
3.10. Definition. A module is called *concrete* if it contains no τ-edges, and *deterministic* if no node has two outgoing edges bearing the same label.

3.11. Lemma. If $g \underline{\leftarrow} g$ then $g \cdot \delta \leftrightarrows_\tau h \cdot \delta$ for some concrete deterministic h.

Proof: Recall that $g \underline{\leftarrow} g$ means that if $r(g) \xrightarrow{\sigma} n_1$ and $r(g) \xrightarrow{\sigma} n_2$ then n_1 and n_2 have the same outgoing paths. We identify all such pairs, thus turning all τ-edges into τ-loops. Knowing that a τ-loop bisimulates with a δ-edge, we replace τ-loops by δ-edges, thus forming h. One easily sees that g and h are nearly bisimilar, i.e. they are τ-bisimilar except that possibly their endnode labels do not match. We resolve this by turning all endnodes into failures: $g \cdot \delta \leftrightarrows h \cdot \delta$. Clearly h is concrete and deterministic. □

3.12. Definition. A communication function $\cdot|\cdot : A \times A \to A$ is called *trijective* if there are three subsets $C, \underline{C}, C^\circ \subseteq A$ and operations $\underline{\cdot}$, $\cdot^\circ$ and $\cdot^{-\circ}$ such that:

i) $C, \underline{C}, C^\circ$ and $\{\delta, \tau\}$ are disjoint;

ii)  commutes (i.e. $\underline{\underline{a}} = a$, $(a^\circ)^{-\circ} = a$, etc.);

iii) $a|b = a^\circ$ if $a \in \mathcal{C} \cup \underline{\mathcal{C}}$ and $b = \underline{a}$,
$= \delta$ otherwise;

iv) $a^\circ = \underline{a} = \delta$ if $a \notin \mathcal{C} \cup \mathcal{C}^\circ$.

The operations $\underline{\cdot}$ etc. are extended to $P(A)$ in the usual way, e.g. $\underline{X} = \{\underline{x} \mid x \in X\} - \{\delta\}$. In fact $\underline{\mathcal{C}}$ and $\mathcal{C}^\circ$ are examples of this usage. Note that in fact $\underline{A} = \underline{\mathcal{C}}$, $\underline{\underline{A}} = \mathcal{C}$ and $A^\circ = \mathcal{C}^\circ$. Also note that trijectivity implies HA.

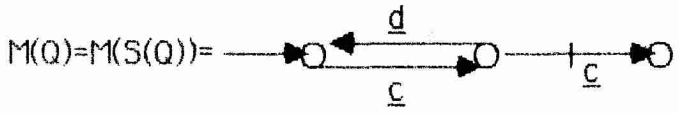
3.13. Fair FIFO queues satisfy an algebraic criterion [5], viz. X is a queue iff $\tau \cdot \partial_H(X \| T) \cdot \delta = \tau \cdot \partial_H(Q \| T) \cdot \delta$ where Q is a standard example of a queue, and H and T are suitably chosen. In other words, X is a queue iff it behaves like one in a suitably chosen test environment. In the sequel we will generalize this idea to a large class of modules. In 3.14 we will define a *tester* T(M) for such modules and in 3.20 we will prove $\tau \cdot \partial_H(X \| T(M)) \cdot \delta = \tau \cdot \partial_H(M \| T(M)) \cdot \delta \Rightarrow X \leftrightarrows M$ modulo some reasonable restrictions.

3.14. Definition. Suppose the communication function is trijective.
Let M be a concrete module.
Now apply the following transformations to M:

1. For each $n \in N(M)$, and for each $a \in \alpha(M)$, if there is neither an edge $n \xrightarrow{a} \cdot$ nor an edge $n \xrightarrow{a}\!\!\!\!\!\rightarrow \cdot$, then add an edge $n \xrightarrow{a} n_0$ to some new terminal node n_0. These extra edges are called *traps* or *trap edges*.
2. Apply $\underline{\cdot}$ to all edge labels.
3. Apply $|\cdot|$ to the graph.

The resulting graph T(M) is called the tester for M.
If S is a specification, T(M(S)) may be abbreviated T(S).

3.15.Examples: i) $S(Q) = ((\underline{c}\ \underline{d})^{\omega}, \{\underline{d}\})$.



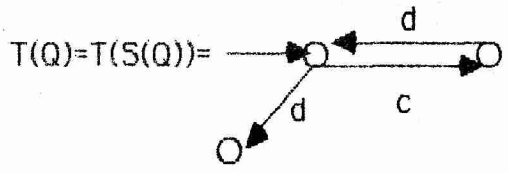
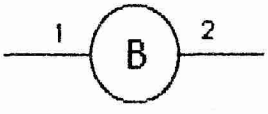


fig3.12

ii) A one-bit buffer $B = (r_1(0)\cdot s_2(0) + r_1(1)\cdot s_2(1))\cdot B$, where port 2 is robust:



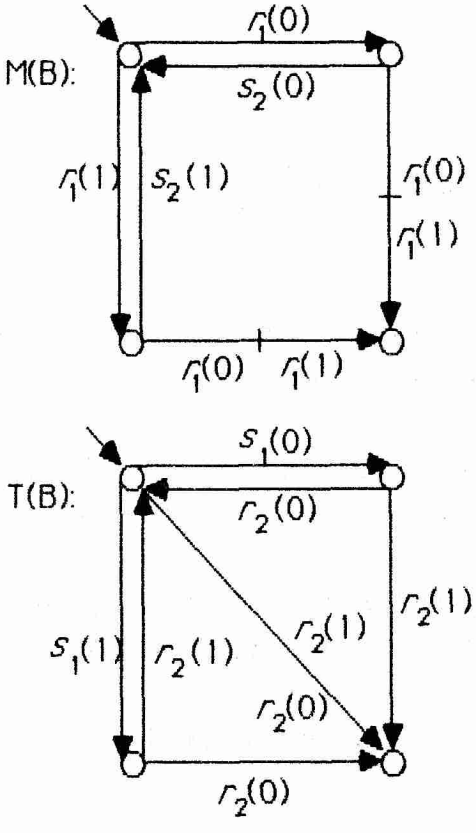


fig3.13

iii) Our running example:

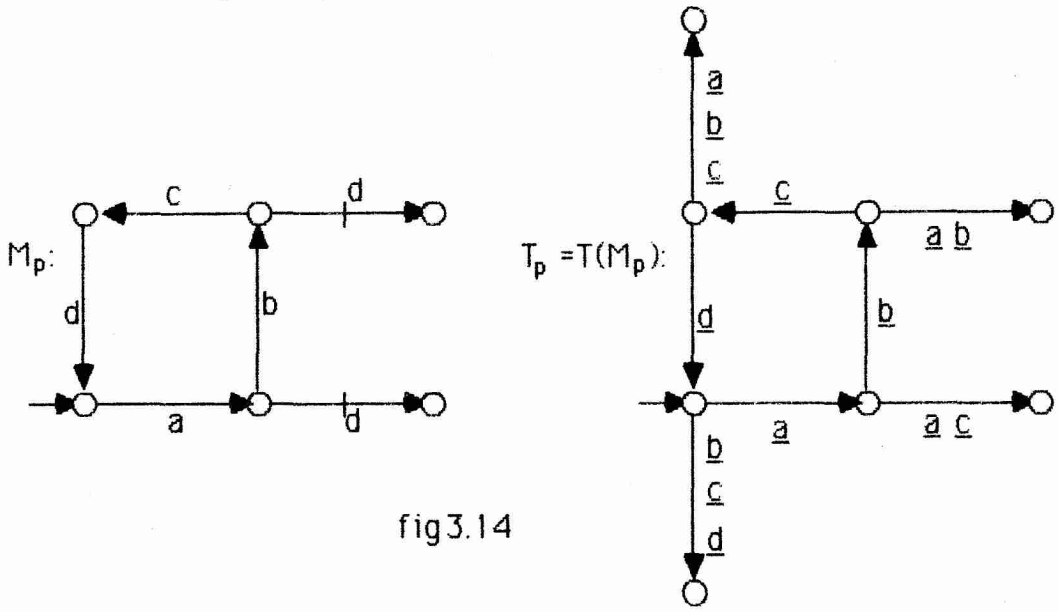


fig 3.14

iv) Our tester for a robust queue over some finite alphabet D would have infinitely many states, indexed by finite sequences over D:

$$T_{\langle\rangle} = \sum_{d \in D} (s_1(d) \cdot T_{\langle d \rangle} + r_2(d))$$

$$T_{\langle d_1,\ldots,d_n \rangle} = r_2(d_1) \cdot T_{\langle d_2,\ldots,d_n \rangle} + \sum_{d_{n+1} \in D} (s_1(d_{n+1}) \cdot T_{\langle d_1,\ldots,d_{n+1} \rangle} + \sum_{d \neq d_1} r_2(d)$$

Note that this tester is different from the one in [5] in that it communicates at both ends of the queue.

3.16. Remark. It is possible to describe the transformation M~~~>T(M) algebraically, but this is not particularly enlightening.

3.17. In order to do anything nontrivial with modules, we need the whole bunch of operations and relations traditionally defined for graphs.

3.17.1. The constructions for $+$, $\cdot$, ∂_H, and τ_I can be copied verbatim from [3].

3.17.2. the constructions for $\|$, $\between$, and $|$ are the usual Cartesian product constructions, augmented with a clause that diagonal edges representing succesful communication are barred iff at least one of the composing edges is.

3.17.3. The definition of $\leftrightarrows_\tau$ is augmented with a clause that barred edges should correspond to barred edges (bearing the same label, of course). This implies that $\xrightarrow{\tau}$ edges can only correspond to $\xrightarrow{\tau}$ edges.

3.18. Examples.

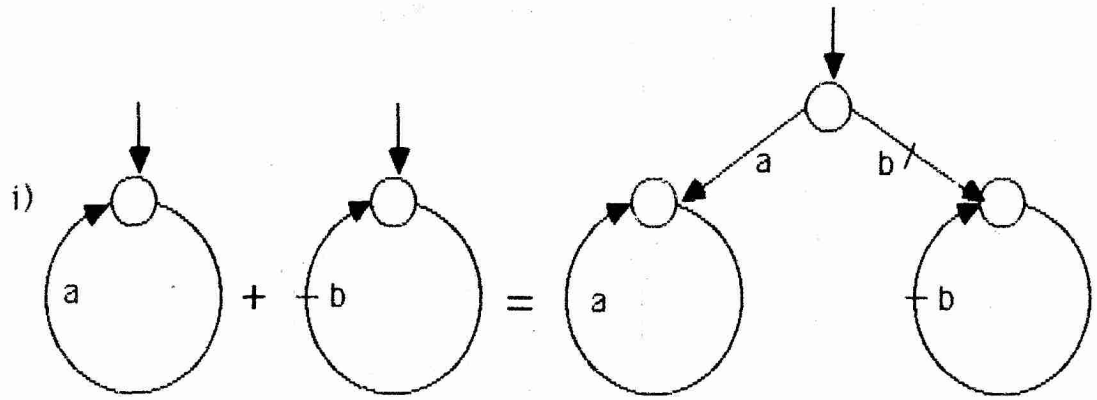


fig3.15

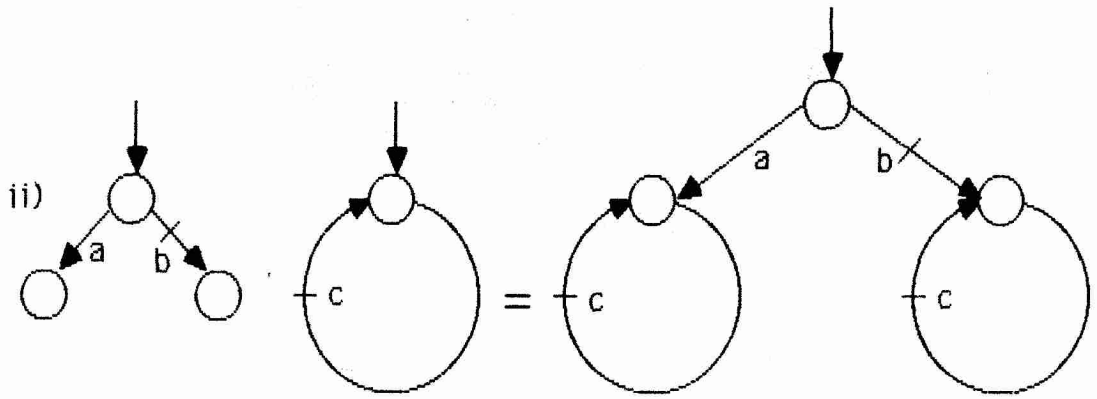


fig3.16

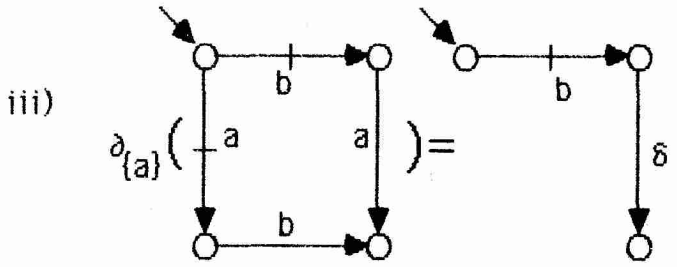


fig3.17

iv)

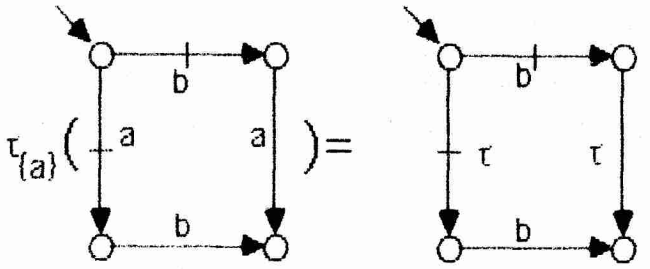


fig3.18

v)

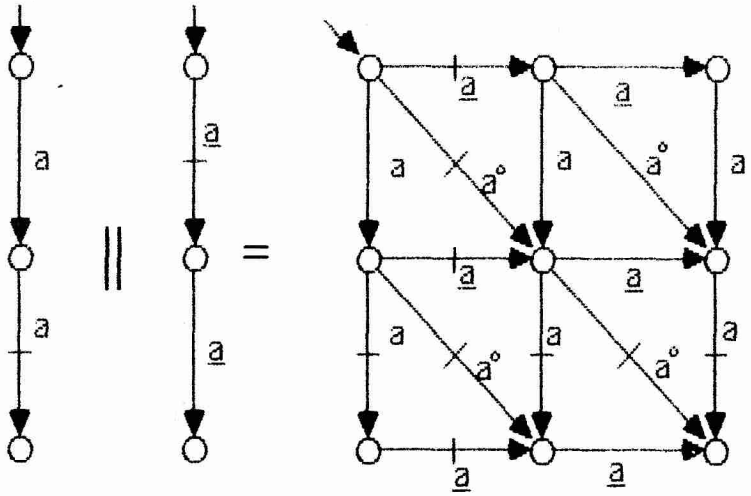


fig3.19

vi) M(Q)‖T(Q) =

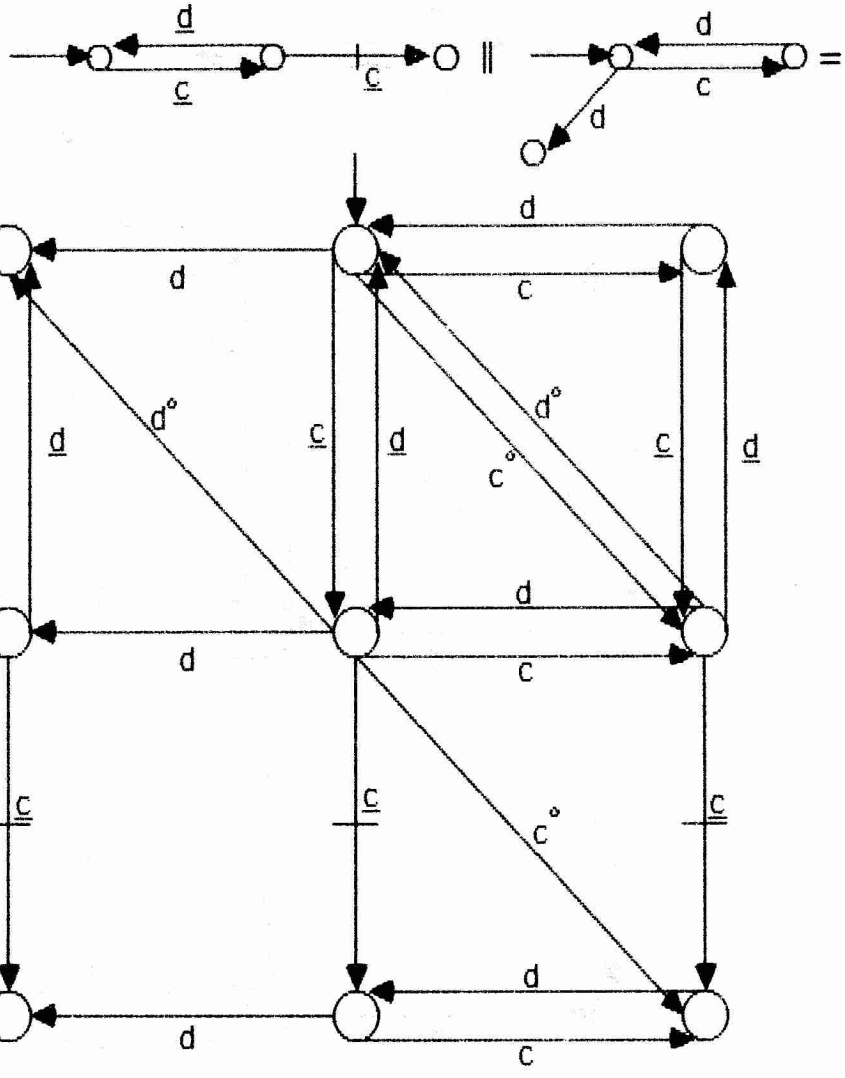


fig 3.20

vii)

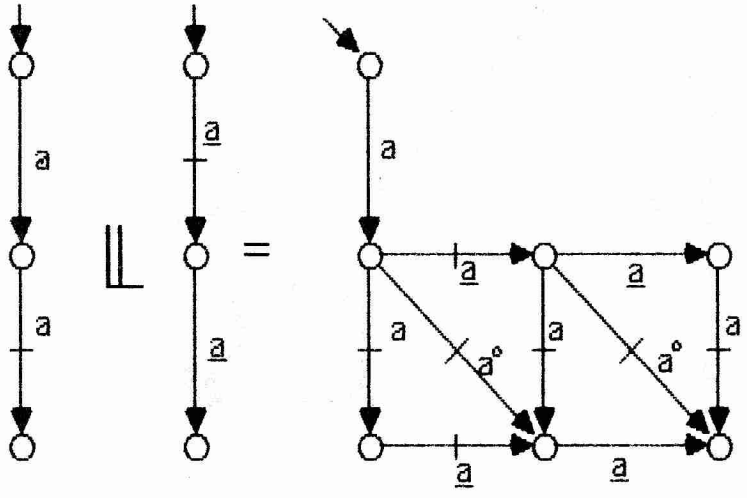


fig 3.21

viii) M(Q)||T(Q) =

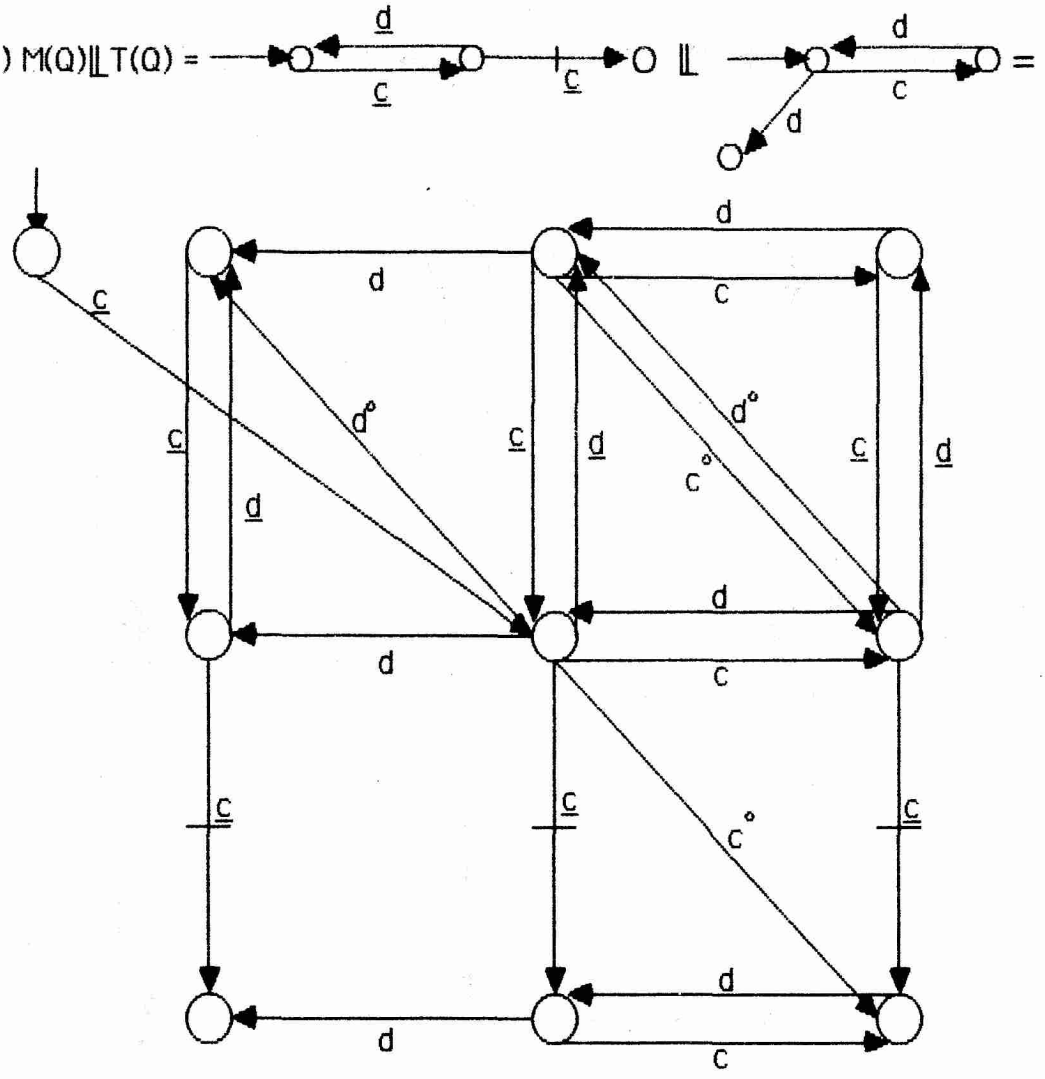


fig 3.22

ix)

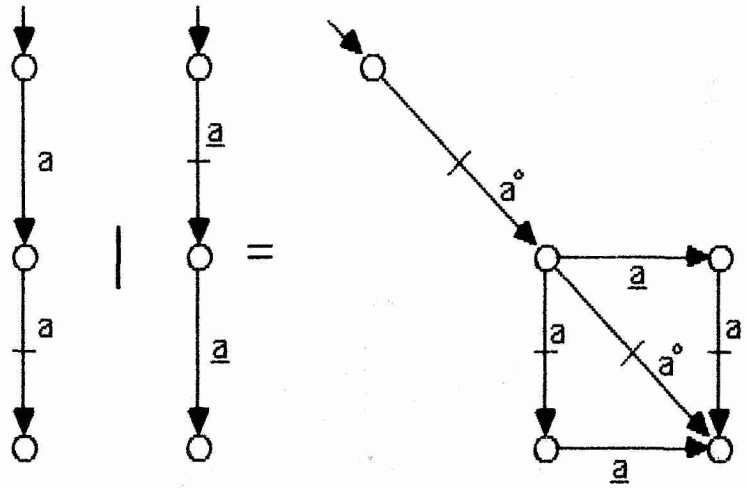
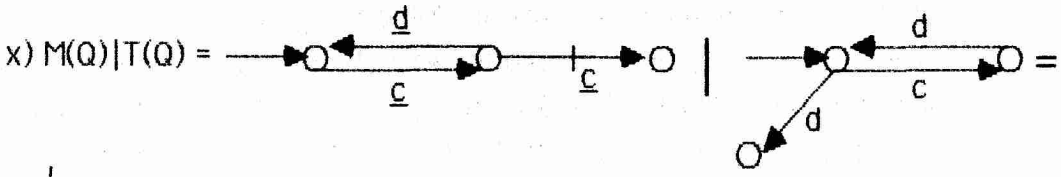


fig 3.23

x) M(Q)|T(Q) =



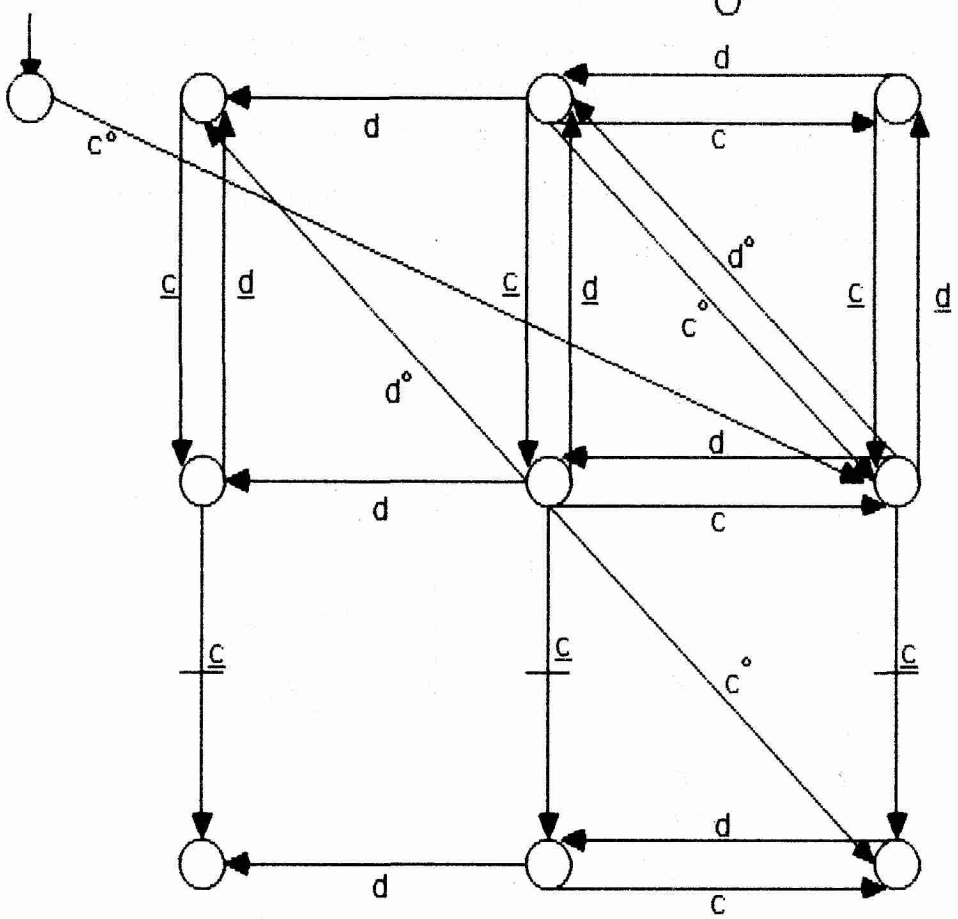


fig 3.24

xi)

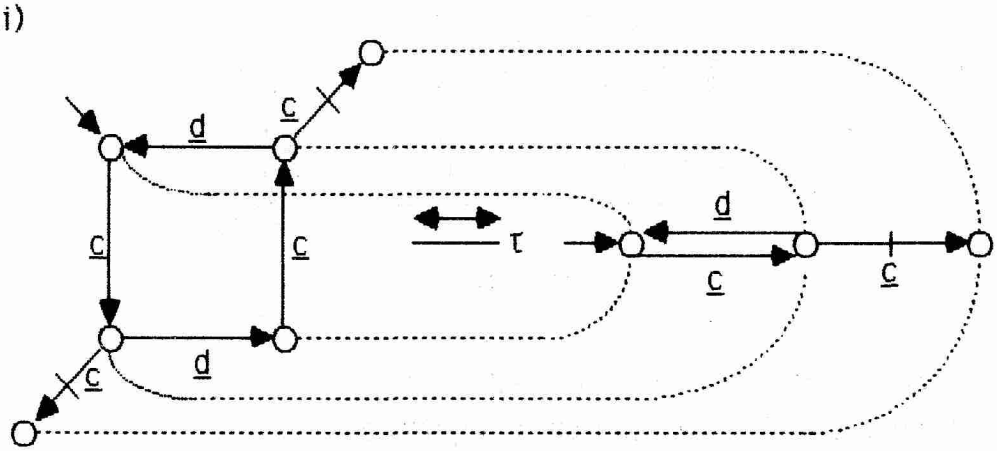


fig 3.25

3.19.1.Lemma. i) |M| || |N| = | M||N |

ii) !M! || !N! $\approx_\tau$! M||N !

Proof: i) Follows trivially from the definitions.

ii) Call a point in the Cartesian product graph *semi-loopy* if at least one of its coordinates is loopy. If ℓ is the loopy node in

!M! then any edge $n \xrightarrow{a} m$ in !N! will give rise to an edge $(\ell,n) \xrightarrow{a} (\ell,m)$, and similarly with N and M reversed. Those and the loops $(\ell,n) \xrightarrow{a} (\ell,n)$ are precisely the outgoing edges of the semi-loopy nodes in !M!||!N!. Consequently, any outgoing path from a semi-loopy node ends in a semi-loopy node (possibly the same), and any sequence of labels can be obtained. Hence we can bisimulate !M!||!N! and !M||N! by relating the semi-loopy points in !M!||!N! to the loopy point in !M||N! and the rest to "themselves". □

3.19.2. Example:

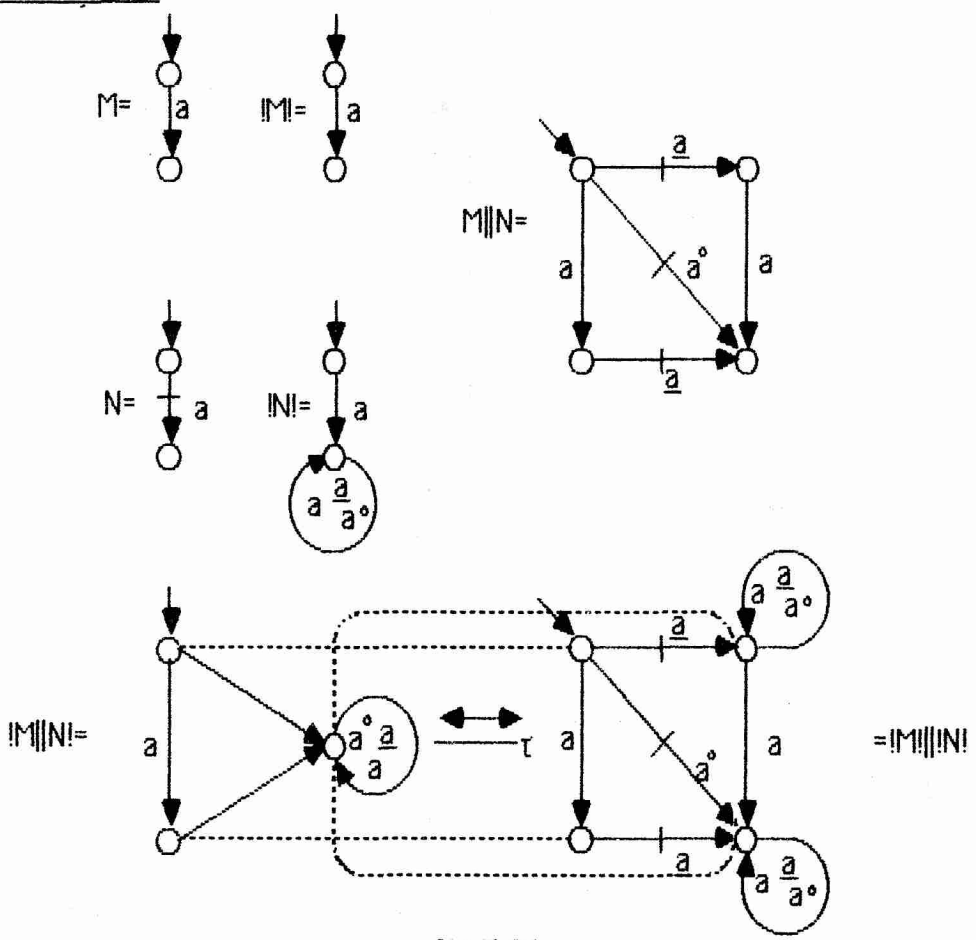


fig 3.26

3.20. Theorem. (TEster Principle, or TEP)

Let M be a module, T=T(M) its tester, X any process.

Suppose: 1. Communication is trijective.

2. M is concrete and deterministic.

3. $\alpha(X)=\alpha(M)\subseteq \mathcal{C}=\underline{\mathcal{A}}$ i.e. X and M consist of communication actions only.

4. $\partial_H(X\|T)\cdot\delta \leftrightarrows_\tau \partial_H(M\|T)\cdot\delta$, where $H=\alpha(M)\cup\underline{\alpha(M)}$.

Then X↔M.

Proof: Abbreviate: $L := \partial_H(X\|T)\cdot\delta$, $R := \partial_H(M\|T)\cdot\delta$.

$|X| \leftarrow |M|$: Suppose a path $\pi_M: r(|M|) \xrightarrow{\sigma} m_1$ can be lengthened to $\pi'_M: r(|M|) \xrightarrow{\sigma} m_1 \xrightarrow{\sigma'} m_2$ and $|X|$ also contains a path $\pi_X: r(|X|) \xrightarrow{\sigma} x_1$. $|M|$ being concrete, we may assume w.l.o.g. that σ' is in fact a singleton $\langle a \rangle$. By constuction, T contains a path $\pi'_T: r(T) \xrightarrow{\underline{\sigma}} t_1 \xrightarrow{\underline{a}} t_2$. Note that $\underline{a} \neq \delta$ in view of condition 3.
Consequently, there is a path $\pi'_R: r(R) \xrightarrow{\sigma^\circ} r_1 \xrightarrow{a^\circ} r_2$. Moreover this r_1 is unique, for trijectivity implies that the decomposition $\sigma^\circ = \sigma|\underline{\sigma}$ is unique, condition 3 implies that σ can only come from $|M|$ and also that $\alpha(T) \subseteq \underline{\alpha(M)} \subseteq C$, so $\underline{\sigma}$ can only come from T; finally, $|M|$ being deterministic implies that π_M is the unique path labeled σ, and that its tester T is also deterministic, hence $\pi_T: r(T) \xrightarrow{\underline{\sigma}} t_1$ is the unique path labeled $\underline{\sigma}$.

The presence of paths $\pi_X: r(|X|) \xrightarrow{\sigma} x_1$ in $|X|$ and $\pi_T \xrightarrow{\underline{\sigma}} t_1$ in T gives rise to the existence of a path $\pi_L: r(|L|) \xrightarrow{\sigma^\circ} \ell_1$ in L. In the bisimulation relation asserted by $|L| \leftrightarrows_c |R|$, ℓ_1 must be related to r_1 above, because the path $r(|R|) \xrightarrow{\sigma^\circ} r_1$ in $|R|$ is unique.
As $r_1 \xrightarrow{a^\circ} r_2$ there must be a path $\ell_1 \xrightarrow{\langle a^\circ \rangle} \ell_2$. By trijectivity and condition 3 this can only come from a path $x_1 \xrightarrow{\langle a \rangle} x_2$ (and a matching path in T, of course).
So we see that we can lengthen any path $r(|X|) \xrightarrow{\sigma} x_1$ to $r(|X|) \xrightarrow{\sigma} x_1 \xrightarrow{\langle a \rangle} x_2$.

$!M! \leftarrow !X!$: Suppose a path $\pi_x: r(!X!) \xrightarrow{\sigma} x_1$ can be lengthened to $r(!X!) \xrightarrow{\sigma} x_1 \xrightarrow{\sigma'} x_2$. We are to show that any path $r(!M!) \xrightarrow{\sigma} m_1$ can be lengthened to $r(!M!) \xrightarrow{\sigma} m_1 \xrightarrow{\sigma'} m_2$. $!M!$ being deterministic, this is equivalent to showing that $!M!$ contains a path labeled $\sigma_0 = \sigma * \sigma'$. We distinguish two cases:

Case 1: $!L!$ contains a path labeled σ°_0. Then $!R!$ does so, too. Trijectivity and condition 3 now imply that we can "project" this path down to $!M!$.

Case 2: We can find some, possibly empty, initial segment σ°_1 of σ°_0 in $!L!$ (i.e. a path bearing that label), σ°_1 maximal, but shorter than σ°_0. Consider the "projection" onto T:

$r(T) \xrightarrow{\underline{\sigma}} 1 \rightarrow t_1$. Now t_1 cannot be the end node of one of the

trap edges added in step 3.14.1 for those edges do no communicate with edges of M by construction (of T). Neither do trap edges communicate with those of X by condition 4. Consequently, a corrresponding path $r(!M!) \xrightarrow{\sigma_1} m_1$ exists.

Now, since σ_1 is not all of σ_0, the rest of σ_0 must begin with some atom a. So now !X! contains a path $r(!X!) \xrightarrow{\sigma_1} x_1 \xrightarrow{\langle a \rangle} x_3$ and !L! does not contain a similar path $r(!L!) \xrightarrow{\sigma^\circ_1} \ell_1 \xrightarrow{\langle a^\circ \rangle}$. Apparently, T does not contain an edge $t_1 \xrightarrow{a}$. From 3.14. we deduce that this can only happen if M contains an edge $m_1 \xrightarrow{a} m_3$. But then !M! contains an edge $m_1 \xrightarrow{a} m_\ell$, with m_ℓ loopy. Hence we can complete our path $r(!M!) \xrightarrow{\sigma_1} m_1 \xrightarrow{a} m_\ell \xrightarrow{\sigma_2} m_\ell$. □

3.21. Remark. The role of the δ's in condition 4 of 3.20. is not evident from the proof, to say the least. Inspection of the proof shows that the theorem also holds without them. In fact one can derive that result more easily by noting that it is just a weakening of 3.20. Stated differently, condition 4 as given is a weakening of the version without δ's and consequently the theorem as a whole is stronger than the variation.
For example, consider $M = \rightarrow 0 \xrightarrow{a} 0 \xrightarrow{a} 0$, $N = \rightarrow 0 \xrightarrow{\tau} 0 \xrightarrow{a} 0$.
One easily sees that $N \not\models M$.

We calculate: $T(M) = \rightarrow 0 \xrightarrow{a} 0$;

$$L = \partial_{\{a,\underline{a}\}}(M \| T(M)) = \rightarrow 0 \xrightarrow{a^\circ} 0 \xrightarrow{\delta} 0;$$

$$R = \partial_{\{a,\underline{a}\}}(N \| T(M)) = \rightarrow 0 \xrightarrow{\tau} 0 \xrightarrow{a^\circ} 0.$$

We see that L and R are "equal up to leading τ's and trailing δ's". Precisely this relation is expressed by $L \cdot \delta \leftrightarrows_\tau R \cdot \delta$.

3.22. Remark. The reader might wonder why we introduced these testers when after all the relation ⊨ can read off from the graphs directly. However, in our motivating example, CABP, the proposed implementation is not given as a graph, but as a rather complex process term. To draw the graph is of course possible, but we saw in section 2 that it involves a lot of work, and most of it is in fact redundant. The tester equation 3.20.4, on the other hand, encapsulates any redundant edge as soon as possible. That way, we can keep the amount of work feasible. In section 4, we will actually carry out this calculation.

3.23. Lemma. Let $g_i,h_i \in \mathbf{G}$ $(i=1,2)$.

Suppose: i) communication is trijective: $\mathcal{C}|\underline{\mathcal{C}} = \mathcal{C}^\circ$;
ii) $g_i \underline{\leftarrow} h_i$ $(i=1,2)$;
iii) $\alpha(g_1) \subseteq \mathcal{C}$, $\alpha(g_2) \subseteq \underline{\mathcal{C}}$.

Then $g_1 \| g_2 \underline{\leftarrow} h_1 \| h_2$.

Proof: First note that condition ii) implies that $\alpha(h_i) \subseteq \alpha(g_i)$, and hence condition iii) implies $\alpha(h_1) \subseteq \mathcal{C}$, $\alpha(h_2) \subseteq \underline{\mathcal{C}}$.

Next, recall that trijectivity implies that $\mathcal{C}$, $\underline{\mathcal{C}}$ and $\mathcal{C}|\underline{\mathcal{C}}$ are disjoint.

The upshot of all this, is that each trace in $g_1 \| g_2$ (or $h_1 \| h_2$) can be uniquely decomposed into traces in g_1 and g_2 (h_1 and h_2), i.e. we can establish atom by atom where it comes from.

So if a path $\pi: r(h_1 \| h_2) \xrightarrow{\sigma} \cdot$ can be lengthened to $\pi': r(h_1 \| h_2) \xrightarrow{\sigma} \cdot \xrightarrow{\sigma'} \cdot$ we can find the unique labels σ_1, σ_2, σ'_1 and σ'_2 of the (possibly non-unique) paths $\pi_i: r(h_i) \xrightarrow{\sigma_i} \cdot \xrightarrow{\sigma'_i} \cdot$ that "caused" π'. In any case, there exist paths $r(h_i) \xrightarrow{\sigma_i} \cdot$ that can be lengthened to $r(h_i) \xrightarrow{\sigma_i} \cdot \xrightarrow{\sigma'_i} \cdot$.

Moreover, each path $r(g_1 \| g_2) \xrightarrow{\sigma} \cdot$ "originates" from paths $r(g_i) \xrightarrow{\sigma_i} \cdot$ also labeled σ_i. Now $g_i \underline{\leftarrow} h_i$ implies that each such path $r(g_i) \xrightarrow{\sigma_i} \cdot$ can be lengthened to $r(g_i) \xrightarrow{\sigma_i} \cdot \xrightarrow{\sigma'_i} \cdot$, and therefore each path $r(g_1 \| g_2) \xrightarrow{\sigma} \cdot$ can be lengthened to $r(g_1 \| g_2) \xrightarrow{\sigma} \cdot \xrightarrow{\sigma'} \cdot$ as was to be shown. □

3.24. Theorem. (Modular Assembly Principle, or MAP).

Let $M_i, N_i \in \mathbf{M}$ $(i=1,2)$.

Suppose: i) communication is trijective;
ii) $M_i \models N_i$;
iii) $\alpha(N_1) \subseteq \mathcal{C}$, $\alpha(N_2) \subseteq \underline{\mathcal{C}}$.

Then $M_1 \| M_2 \models N_1 \| N_2$.

Proof: $|M_1 \| M_2| \underline{\leftarrow} |N_1 \| N_2|$: This follows directly from 3.19.i and 3.23.

$!M_1 \| M_2! \underline{\leftarrow} !N_1 \| N_2!$: Suppose, as always, a path $\pi_M: r(!M_1 \| M_2!) \xrightarrow{\sigma} \cdot$ can be lengthened to $\pi'_M: r(!M_1 \| M_2) \xrightarrow{\sigma} \cdot \xrightarrow{\sigma'} \cdot$, and consider a path $\pi_N: (!N_1 \| N_2!) \xrightarrow{\sigma} \cdot$. There are three cases:

1. π_N passes through the loopy node of $!N_1 \| N_2!$. In that case the result is trivial, because in a loopy node one can construct

any trace one likes.

2. π_N does not pass through the loopy node, but π_M does. This contradicts the fact that $!N_1!\leftarrow!M_1!$ and $!N_2!\leftarrow!M_2!$, because for entering a loopy node in the merge, one of the components has to go through a barred edge, thereby entering a loopy node itself.
3. π_M and π_N do not pass through loopy nodes. In this case the proof of 3.23 works, using the observation that when π'_M enters the loopy node in $!M_1\|M_2!$ we can choose to continue this path in that component that caused the loopy node (and hence has a loopy node itself. □

3.25. Recall that barred edges are supposed to be redundant, i.e. we intended them to occur in a context in which they cannnot communicate. So if we encapsulate we expect all barred edges to disappear. Consequently, the resulting graphs should be robust. As was to be expected, this is the case in our motivating example, CABP:

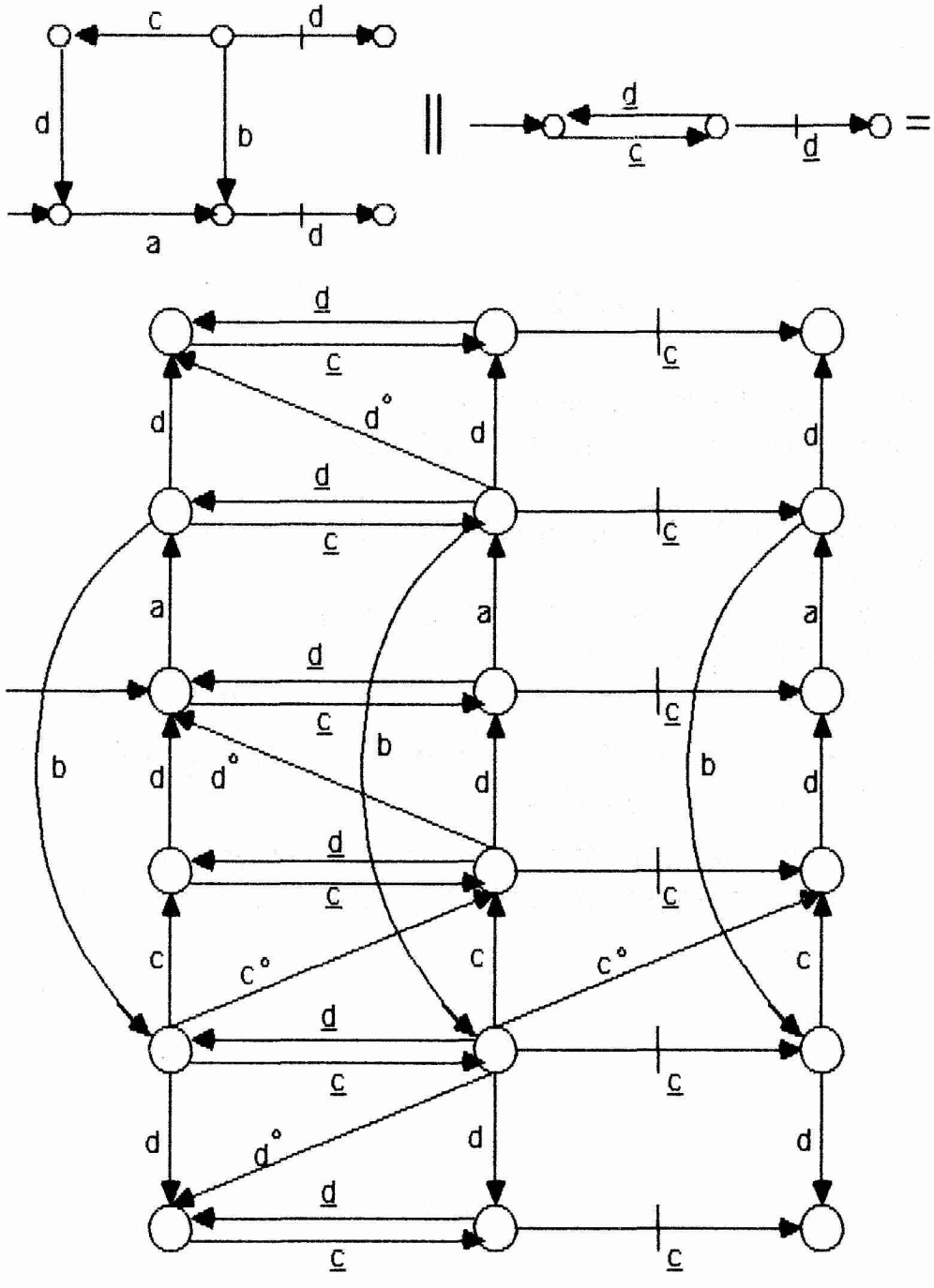


fig 3.27

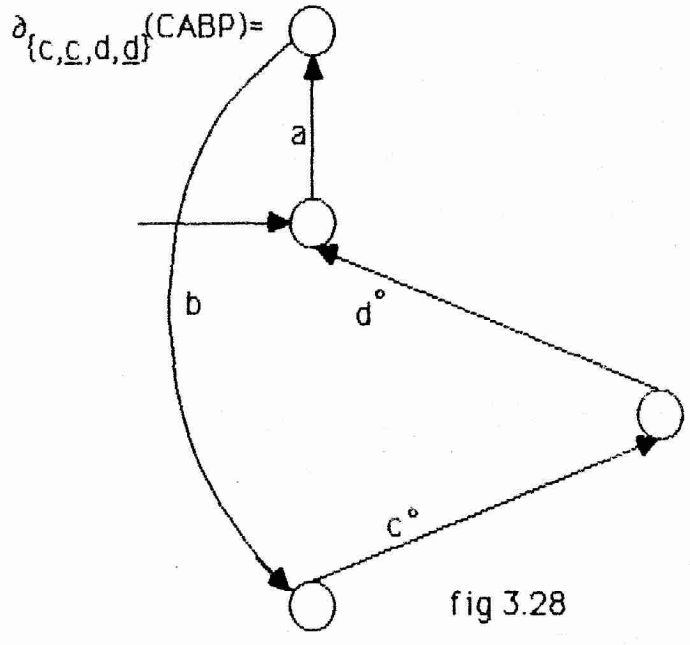


fig 3.28

3.26. Theorem. (RObustness Principle, or ROP).
If g⊨h, and both g and H are robust, then τ·g·δ = τ·h·δ.

Proof: Robustness means that |g|=!g!=g and |h|=!h!=h. So g⊨h boils down to g←h and h←g.
Now define a relation R on N(g)xN(h) by putting
$nRk :\Leftrightarrow \exists\sigma[(\exists\pi{:}r(g)\xrightarrow{\sigma}\!\!\!\!\rightarrow n) \wedge (\exists\pi'{:}r(h)\xrightarrow{\sigma}\!\!\!\!\rightarrow k)]$.
Then, if n_1Rk_1 and there is a path $n_1\xrightarrow{\sigma'}\!\!\!\!\rightarrow n_2$ in g, then h←g implies that there exists a path $k_1\xrightarrow{\sigma'}\!\!\!\!\rightarrow k_2$ in h, and so n_2Rk_2, for $r(g)\xrightarrow{\sigma*\sigma'}\!\!\!\!\rightarrow n_2$ and $r(h)\xrightarrow{\sigma*\sigma'}\!\!\!\!\rightarrow k_2$.
The same holds with g and h interchanged. In other words, g and h are τ-bisimilar, except possibly for their end-node labels. I.e., τ·g·δ=τ·h·δ. □

4. A Verification of CABP.

4.1. Warning: This section contains nothing but boring calculations. In 4.3. through 4.6. TEP is applied to C‖L‖D, in 4.7. through 4.10. it is applied to A‖K‖B, and, finally, in 4.11. an algebraic reformulation of 3.25. follows.

4.2. notation. In 1.3. and 2.2. we named some subsets of A:

$H = \{r(d,p), s(d,p) \mid d \in \mathbf{D}, p \in \{2,3,5,6,7,8\}\}$

$I = \{t(d,p) \mid d \in \mathbf{D}, 1 \leq p \leq 8\} \cup \{ik, jk, kk, il, jl, kl\}$

$H_P = \{r(d,p), s(d,p) \mid d \in \mathbf{D}, p \in \{2,3\}\}$

$I_P = \{t(d,p) \mid d \in \mathbf{D}, p \in \{2,3\}\} \cup \{ik, jk, kk\}$

$H_Q = \{r(d,p), s(d,p) \mid d \in \mathbf{D}, p \in \{6,7\}\}$

$I_Q = \{t(d,p) \mid d \in \mathbf{D}, p \in \{6,7\}\} \cup \{il, jl, kl\}$

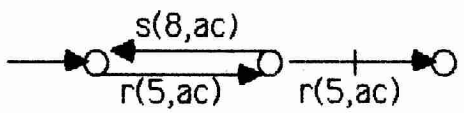
We defined $P = \tau_{I_P}\partial_{H_P}(A\|K\|B)$ and $Q = \tau_{I_Q}\partial_{H_Q}(C\|L\|D)$ and noticed that $CA1,2,5,6,7 \vdash \tau_I\partial_H(A\|K\|B\|C\|L\|D) = \tau_I\partial_H(P\|Q)$.

In order to keep the text readable, we will refrain from using *italics* from here on, we will simply write r(d,p) etc.

4.3. In the next four subsections we will verify that $Q \leftrightarrows M(Q)$.

According to TEP (3.20.), it suffices to show that

$$(*) \quad \tau \cdot \partial_{HQ}(Q\|M(Q)) \cdot \delta = \tau \cdot \partial_{HQ}(M(Q)\|T(M(Q))) \cdot \delta$$

Recall that M(Q)= 

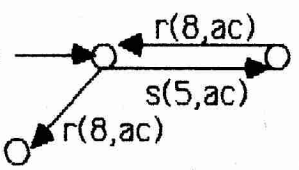
and T(Q)=T(M(Q))= 

fig 4.1

In order to calculate M(Q)‖T(M(Q)) we need some notation for the barred r(5,ac)-edge of M(Q). We choose r(5,ac)·χ, inspired by 2.2, where barred edges were introduced to denote "r(5,ac), followed by some unidentified process". More formally, barred edges communicate and are

encapsulated like their unbarred counterparts. At any rate, χ should be eliminable anyway.

4.4. We will abbreviate: $5 = s(5,ac)$, $\underline{5} = r(5,ac)$, $5^\circ = t(5,ac)$,
$8 = s(8,ac)$, $\underline{8} = r(8,ac)$, $8^\circ = t(8,ac)$,
$T_1 = T = T(M(Q))$, $T2 = \underline{8}\cdot T_1$,
$M_1 = M(Q)$, $M_2 = 8\cdot M_1 + 5\cdot\chi$.

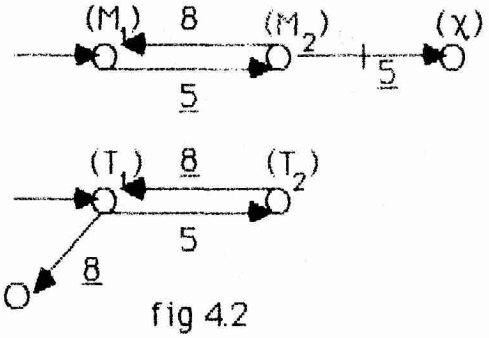


fig 4.2

So: $T_1 = 5\cdot T2 + \underline{8}$,
$T_2 = \underline{8}\cdot T_1$,
$M_1 = \underline{5}\cdot M_2$,
$M_2 = 8\cdot M_1 + 5\cdot\chi$.

Hence $T_1\|M_1 = 5\cdot(T_2\|M_1) + \underline{8}\cdot M_1 + \underline{5}\cdot(T_1\|M_2) + 5^\circ\cdot(T_2\|M_2)$

and $T_2\|M_2 = \underline{8}\cdot(T_1\|M_2) + 8\cdot(T_2\|M_1) + \underline{5}\cdot(T_2\|\chi) + 8^\circ\cdot(T_1\|M_1)$.

Consequently $\partial_{(5,\underline{5},8,\underline{8})}(T_1\|M_1) = 5^\circ\cdot\partial_{(5,\underline{5},8,\underline{8})}(T_2\|M_2)$

and $\partial_{(5,\underline{5},8,\underline{8})}(T_2\|M_2) = 8^\circ\cdot\partial_{(5,\underline{5},8,\underline{8})}(T_1\|M_1)$.

Therefore $\partial_{HQ}(T_1\|M_1) = 5^\circ\cdot 8^\circ\cdot\partial_{HQ}(T_1\|M_1)$.

So $\partial_{HQ}(T_1\|M_1) = (5^\circ\cdot 8^\circ)^\omega$.

So the RHS of (*) reduces to $\tau\cdot(5^\circ\cdot 8^\circ)^\omega\cdot\delta$.

4.5. In order to investigate $Q = \tau_{IQ}\partial_{HQ}(C\|L\|D)$, we will need quite a lot of notation. Firstly, we abbreviate:

$5 = s(5,ac)$, $6_b = s(6,b)$ $(b=0,1)$, $7_b = s(7,b)$ $(b=0,1,e)$, $8 = s(8,ac)$,
$\underline{5}, \underline{6}_b, \underline{7}_b, \underline{8} = r(\cdot,\cdot)$, $5^\circ, 6^\circ_b, 7^\circ_b, 8^\circ = t(\cdot,\cdot)$.

Next, we need names for all states of the processes involved:

C:

$$C = C^1$$

$$C^b = \underline{5} \cdot C^{1-b} + 6_b \cdot C^b$$

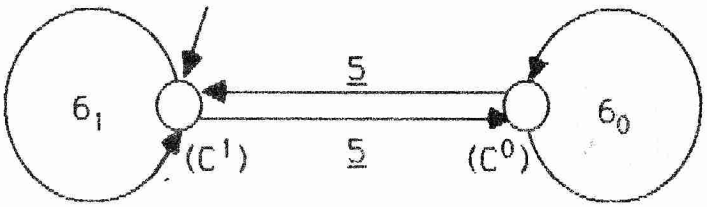


fig 4.3

L:

$$L = L_1$$

$$L_1 = \underline{6_0} \cdot L^0{}_2 + \underline{6_1} \cdot L^1{}_2$$

$$L^b{}_2 = i1 \cdot L^b{}_3 + j1 \cdot L_4 + k1 \cdot L_1$$

$$L^b{}_3 = 7_b \cdot L_1$$

$$L_4 = 7_e \cdot L_1$$

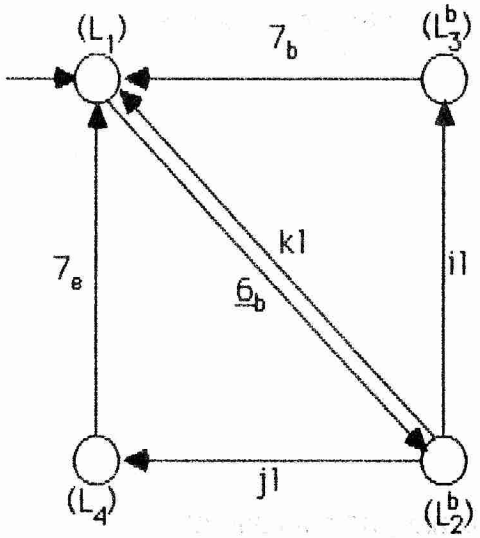


fig 4.4

D:

$$D = D^0{}_1$$

$$D^b{}_1 = \underline{Z}_b \cdot D^b{}_2 + (\underline{Z}_{1-b} + \underline{Z}_e) \cdot D^b{}_1$$

$$D^b{}_2 = 8 \cdot D^{1-b}{}_1$$

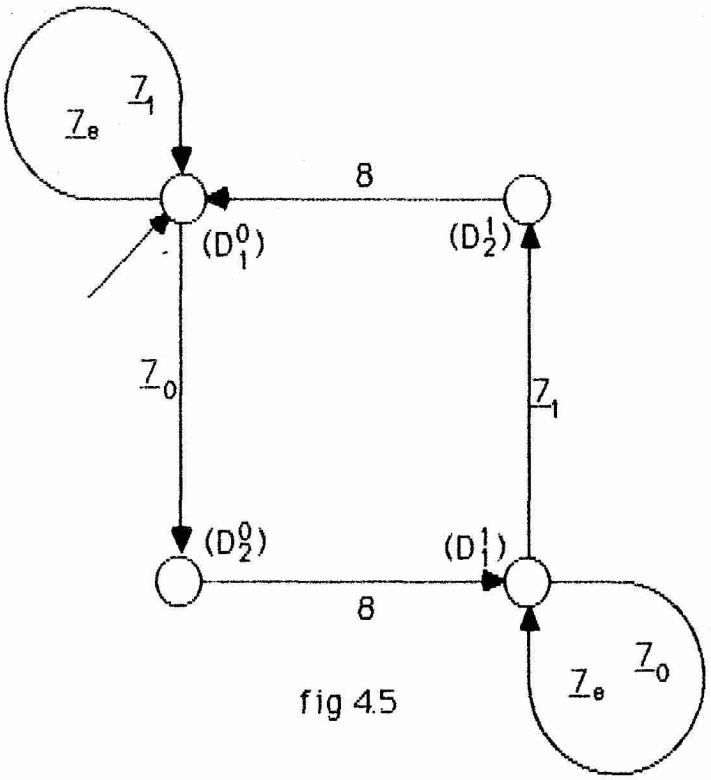


fig 4.5

Finally, we introduce names for some relevant composite terms:

$$Q_{1,b,i} = \partial_H (T_1 \parallel C^{1-b} \parallel L^{(1-b)}{}_i \parallel D^b{}_1)$$

$$Q_{2,b,i} = \partial_H (T_2 \parallel C^b \parallel L^{(1-b)}{}_i \parallel D^b{}_1)$$

$$Q_{3,b,i} = \partial_H (T_2 \parallel C^b \parallel L^{(b)}{}_i \parallel D^b{}_1)$$

$$Q_{4,b,i} = \partial_H (T_2 \parallel C^b \parallel L^{(b)}{}_i \parallel D^b{}_2)$$

where b=0,1, i=1,...,4 and (b) is either b or blank, as appropriate.

4.6. Claim: $\partial_H(T\|Q) = \tau\cdot(5^\circ\cdot 8^\circ)^\omega$.

4.6.1. Claim: $\partial_H(T\|Q) = \tau_{IQ}(Q_{1,0,1})$.

Proof:
$$\begin{aligned}
\partial_H(T\|Q) &= \partial_H\,(T\|\tau_{IQ}\partial_{HQ}(C\|L\|D)) \\
&= \partial_H\tau_{IQ}\,(T\|\tau_{IQ}\partial_{HQ}(C\|L\|D)) && \text{by CA4} \\
&= \partial_H\tau_{IQ}\,(T\|\partial_{HQ}(C\|L\|D)) && \text{by CA2} \\
&= \tau_{IQ}\partial_H\,(T\|\partial_{HQ}(C\|L\|D)) && \text{by CA7} \\
&= \tau_{IQ}\partial_H\,(T\|C\|L\|D) && \text{by CA1} \\
&= \tau_{IQ}\partial_H\,(T_1\|C^{1-0}\|L_1\|D^0{}_1) \\
&= \tau_{IQ}\,(Q_{1,0,1}) && \square
\end{aligned}$$

4.6.2. Claim: $\tau_{IQ}(Q_{1,b,i}) = \tau\cdot\sum_{i=1}^{4} 5^\circ\cdot\ \tau_{IQ}\,(Q_{2,b,i})$.

Proof:
$$\begin{aligned}
Q_{1,b,1} &= 5^\circ\cdot Q_{2,b,1} + 6^\circ_{1\text{-}b}\cdot Q_{1,b,2} \\
Q_{1,b,2} &= 5^\circ\cdot Q_{2,b,2} + il\cdot Q_{1,b,3} + jl\cdot Q_{1,b,4} + kl\cdot Q_{1,b,1} \\
Q_{1,b,3} &= 5^\circ\cdot Q_{2,b,3} + 7^\circ_{1\text{-}b}\cdot Q_{1,b,1} \\
Q_{1,b,4} &= 5^\circ\cdot Q_{2,b,4} + 7^\circ_{e}\cdot Q_{1,b,1}
\end{aligned}$$

We find that $\{Q_{1,b,i}\mid i=1,\ldots,4\}$ is a cluster. Applying CFAR yields:

$$\begin{aligned}
\tau_{IQ}(Q_{1,b,i}) &= \tau\cdot\tau_{IQ}\Big(\sum_{i=1}^{4} 5^\circ\cdot Q_{2,b,i}\Big) \\
&= \tau\cdot\sum_{i=1}^{4} 5^\circ\cdot\ \tau_{IQ}(Q_{2,b,i}) \qquad \square
\end{aligned}$$

4.6.3. Claim: $\tau_{IQ}(Q_{2,b,i}) = (\tau\cdot)\ \tau_{IQ}(Q_{3,b,1})$

Proof:
$$\begin{aligned}
&Q_{2,b,1} = \partial_H\,(T_2\|C^b\|L_1\|D^b{}_1) = Q_{3,b,1} \\
&Q_{2,b,3} = 7^\circ_{1\text{-}b}\cdot Q_{3,b,1} && \text{hence } \tau_{IQ}(Q_{2,b,3}) = \tau\cdot\tau_{IQ}(Q_{3,b,1}) \\
&Q_{2,b,4} = 7^\circ_{e}\cdot Q_{3,b,1} && \text{hence } \tau_{IQ}(Q_{2,b,4}) = \tau\cdot\tau_{IQ}(Q_{3,b,1}) \\
&Q_{2,b,2} = il\cdot Q_{2,b,3} + jl\cdot Q_{2,b,4} + kl\cdot Q_{2,b,1} \\
&\qquad\qquad \text{hence } \tau_{IQ}(Q_{2,b,3}) = (\tau^2+\tau^2+\tau)\cdot\ \tau_{IQ}(Q_{3,b,1})
\end{aligned}$$

The claim follows. $\square$

4.6.4. Claim: $\tau_{I_Q}(Q_{3,b,1}) = \tau \cdot \tau_I(Q_{4,b,1})$.

Proof: $Q_{3,b,1} = 6^\circ_{1-b} \cdot Q_{3,b,2}$

$Q_{3,b,2} = il \cdot Q_{3,b,3} + jl \cdot Q_{3,b,4} + kl \cdot Q_{3,b,1}$

$Q_{3,b,4} = 7^\circ_e \cdot Q_{3,b,1}$

We find that $\{Q_{3,b,i} \mid i=1,2,4\}$ is a cluster. Applying CFAR yields:

$$\begin{aligned}\tau_{I_Q}(Q_{3,b,1}) &= \tau \cdot \tau_{I_Q}(il \cdot Q_{3,b,3}) \\ &= \tau \cdot \tau_{I_Q}(il \cdot 7^\circ_b \cdot Q_{4,b,1}) \\ &= \tau^3 \cdot \tau_{I_Q}(Q_{4,b,1}) \\ &= \tau \cdot \tau_{I_Q}(Q_{4,b,1}) \quad \square\end{aligned}$$

4.6.5. Claim: $\tau_{I_Q}(Q_{4,b,1}) = \tau \cdot \sum_{i=1}^{4} (\tau\cdot)\, 8^\circ \cdot Q_{1,1-b,i}$. Here $(\tau\cdot)$ is $\tau\cdot$ iff i=3,4.

Proof: $Q_{4,b,1} = 8^\circ \cdot Q_{1,b-1,1} + 6^\circ_b \cdot Q_{4,b,2}$

$Q_{4,b,2} = 8^\circ \cdot Q_{1,b-1,2} + il \cdot Q_{4,b,3} + jl \cdot Q_{4,b,4} + kl \cdot Q_{4,b,1}$

So $Q_{4,b,1}$ and $Q_{4,b,2}$ form a cluster and (using CFAR):

$$\begin{aligned}\tau_{I_Q}(Q_{4,b,1}) &= \tau \cdot \tau_{I_Q}(8^\circ \cdot Q_{1,b-1,1} + 8^\circ \cdot Q_{1,b-1,2} + il \cdot Q_{4,b,3} + jl \cdot Q_{4,b,4}) \\ &= \tau \cdot \tau_{I_Q}(8^\circ \cdot Q_{1,b-1,1} + 8^\circ \cdot Q_{1,b-1,2} + il \cdot 8^\circ \cdot Q_{1,1-b,3} + jl \cdot 8^\circ \cdot Q_{1,1-b,4}) \\ &= \tau \cdot (8^\circ \cdot \tau_{I_Q}(Q_{1,b-1,1}) + 8^\circ \cdot \tau_{I_Q}(Q_{1,b-1,2}) + \tau \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,3}) + \tau \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,4})) \\ &= \tau \cdot \sum_{i=1}^{4} (\tau\cdot)\, 8^\circ \cdot Q_{1,1-b,i} \quad \square\end{aligned}$$

4.6.6. Summing up:

$$\tau_{I_Q}(Q_{1,b,i}) = \tau \cdot \sum_{i=1}^{4} 5^\circ \cdot \tau_{I_Q}(Q_{2,b,i}) = \tau \cdot \sum_{i=1}^{4} 5^\circ \cdot (\tau\cdot)\, \tau_{I_Q}(Q_{3,b,1}) = \tau \cdot 5^\circ \cdot \tau \cdot \tau_{I_Q}(Q_{3,b,1})$$

$$= \tau \cdot 5^\circ \cdot \tau \cdot \tau_{I_Q}(Q_{4,b,1}) = \tau \cdot 5^\circ \cdot \tau \cdot \tau \cdot \sum_{i=1}^{4} (\tau\cdot)\, 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,i}) =$$

$$= \tau \cdot 5^\circ \cdot \sum_{i=1}^{4} (\tau\cdot)\, 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,i}).$$

We find that $\tau_{I_Q}(Q_{1,1-b,i}) = \ldots = \tau \cdot 5^\circ \cdot \tau_{I_Q}(Q_{3,1-b,1})$.
Hence $\sum_i (\tau\cdot)\, 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,i}) = \sum_i (\tau\cdot)\, 8^\circ \cdot \tau \cdot 5^\circ \cdot \tau_{I_Q}(Q_{3,1-b,1})$
$= \tau \cdot 8^\circ \cdot 5^\circ \cdot \tau_{I_Q}(Q_{3,1-b,1}) = \tau \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,1})$, say.
Consequently, $\tau_{I_Q}(Q_{4,b,1}) = \tau \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,1})$
and hence $\tau_{I_Q}(Q_{1,b,1}) = \tau \cdot 5^\circ \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1-b,1})$.
Similarly, $\tau_{I_Q}(Q_{1-b,b,1}) = \tau \cdot 5^\circ \cdot 8^\circ \cdot \tau_{I_Q}(Q_{1,1,1})$.
Hence both equal the unique solution of $X = \tau \cdot 5^\circ \cdot 8^\circ \cdot X$, which is $\tau\cdot(5^\circ\cdot 8^\circ)^\omega$.

So the LHS of (*), being equal to $\tau\cdot\tau_{I_Q}(Q_{1,0,1})\cdot\delta$, equals the RHS. This proves (*). □

4.7. In 3.4. we introduced two modules M(P) and M_p differing by one barred edge. For no specific reason, we choose to verify $P \not\equiv M_p$ in the next four subsections. They will be rather similar to the last three.

According to TEP, it suffices to show

(*) $\tau \cdot \partial_H(P \| T(M_p)) \cdot \delta = \tau \cdot \partial_H(M_p \| T(M_p)) \cdot \delta$.

4.8. Notation. We will use abbreviations similar to those in 4.5, e.g. $2_{d,b} = s(2,(d,b))$. Incidentally 5, $\underline{5}$ etc. have the same meaning as in 4.4 whereas T_1 etc. have not.

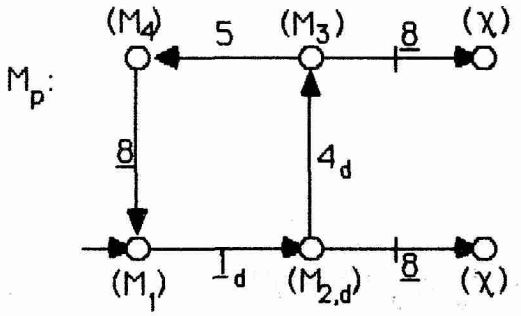


fig 4.6

$M_p = M_1 = \sum_{d\in D} 1_d \cdot M_{2,d}$

$M_{2,d} = 4_d \cdot M_3 + \underline{8} \cdot \chi$
$M_3 = 5 \cdot M_4 + \underline{8} \cdot \chi$
$M_4 = \underline{8} \cdot M_1$

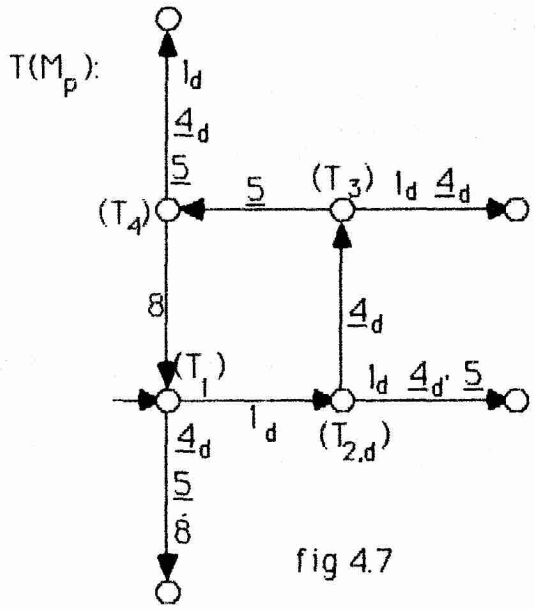


fig 4.7

$$T(M_p) = T_1 = \sum_{d\in D} 1_d \cdot T_{2,d} + \sum_{d\in D} \underline{4}_d + \underline{5} + 8$$

$$T_{2,d} = \underline{4}_d \cdot T_3 + \sum_{d\in D} 1_d + \sum_{d\neq d'} \underline{4}_{d'} + \underline{5}$$

$$T_3 = \underline{5} \cdot T_4 + \sum_{d\in D} 1_d + \sum_{d\in D} \underline{4}_d$$

$$T_4 = 8 \cdot T_1 + \sum_{d\in D} 1_d + \sum_{d\in D} \underline{4}_d + \underline{5}$$

One easily sees:

$$\partial_H(T_1 \| M_1) = \sum_{d\in D} 1^\circ_d \cdot \partial_H(T_{2,d} \| M_{2,d})$$

$$\partial_H(T_{2,d} \| M_{2,d}) = 4^\circ_d \cdot \partial_H(T_3 \| M_3)$$

$$\partial_H(T_3 \| M_3) = 5^\circ \cdot \partial_H(T_4 \| M_4)$$

$$\partial_H(T_4 \| M_4) = 8^\circ \cdot \partial_H(T_1 \| M_1)$$

Therefore the RHS of (*) reduces to $\tau \cdot (\sum_{d\in D} 1^\circ_d \cdot 4^\circ_d \cdot 5^\circ \cdot 8^\circ)^\omega$.

4.9. Like in 4.5., we repeat the relevant parts of 1.4:

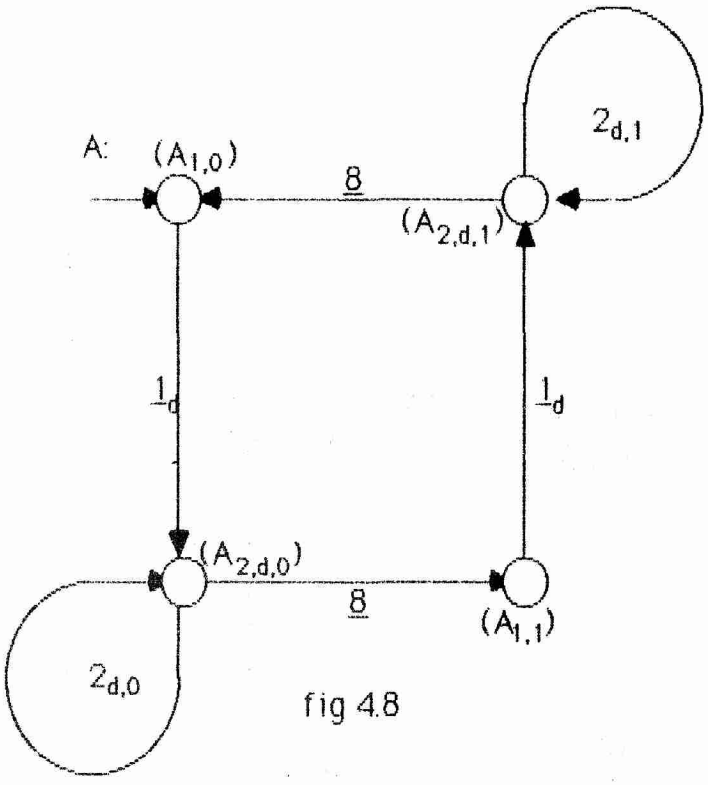


fig 4.8

$$A = A_{1,0}$$

$$A_{1,b} = \sum_{d \in D} \underline{1}_d \cdot A_{2,d,b}$$

$$A_{2,d,b} = 2_{d,b} \cdot A_{2,d,b} + \underline{8} \cdot A_{1,1-b}$$

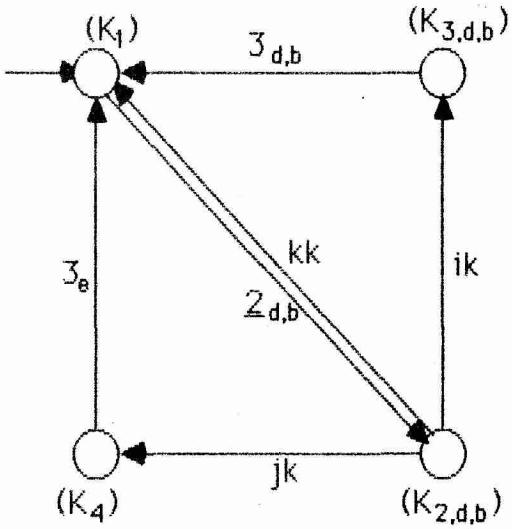


fig 4.9

$$K = K_1 = \sum_{d \in D, b \in B} \underline{2}_{d,b} \cdot K_{2,b,d}$$

$$K_{2,b,d} = ik \cdot K_{3,b,d} + jk \cdot K_4 + kk \cdot K_1$$
$$K_{3,b,d} = \underline{3}_{b,d} \cdot K_1$$
$$K_4 = \underline{3}_e \cdot K_1$$

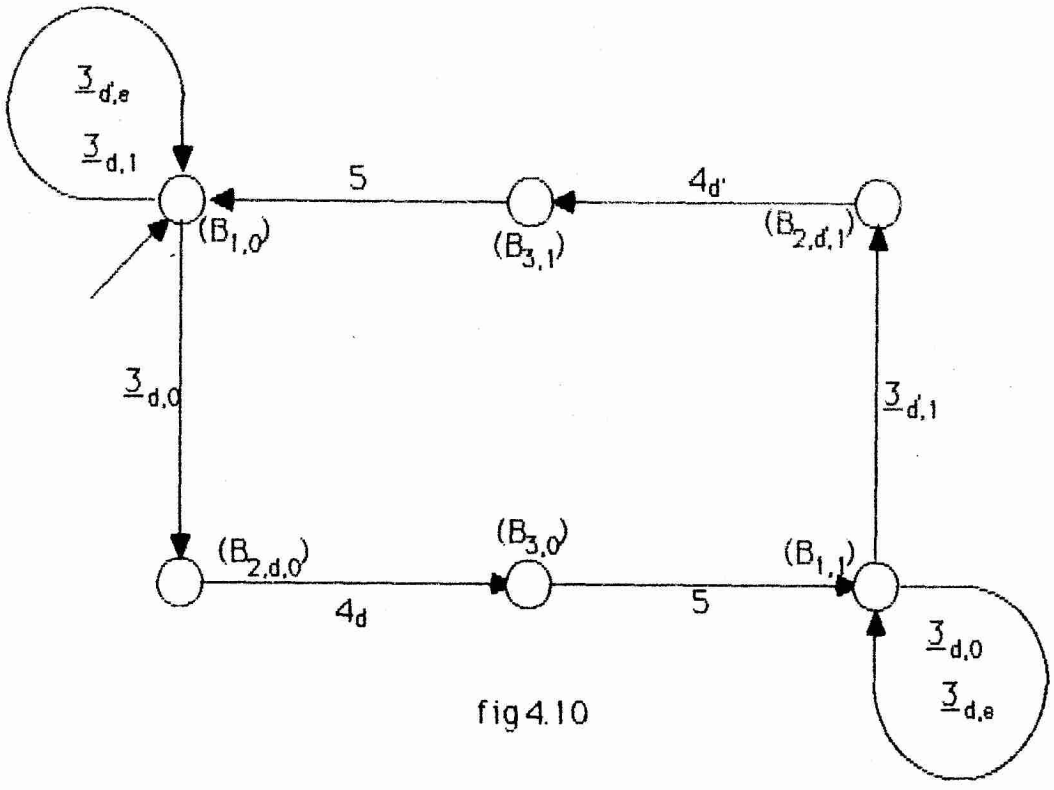


fig 4.10

$$B = B_{1,0}$$
$$B_{1,b} = \sum_{d \in D} \underline{3}_{d,b} \cdot B_{2,d,b} + \sum_{d \in D} \underline{3}_{d,1-b} \cdot B_{1,b} + \underline{3}_e \cdot B_{1,b}$$
$$B_{2,d,b} = 4_d \cdot B_{3,b}$$
$$B_{3,b} = 5 \cdot B_{1,1-b}$$

Relevant composite terms are:

$$P_{1,b,i,(d)} = \partial_H(T_1 \| A_{1,b} \| K_{i,(1-b),(d)} \| B_{1,b})$$

$$P_{2,b,i,d,(d')} = \partial_H(T_{2,d} \| A_{2,d,b} \| K_{i,(1-b),(d')} \| B_{1,b})$$

$$P_{3,b,i,(d)} = \partial_H(T_{2,d} \| A_{2,d,b} \| K_{i,(b),(d)} \| B_{1,b})$$

$$P_{4,b,i,(d)} = \partial_H(T_{2,d} \| A_{2,d,b} \| K_{i,(b),(d)} \| B_{2,d,b})$$

$$P_{5,b,i,(d)} = \partial_H(T_3 \| A_{2,d,b} \| K_{i,(b),(d)} \| B_{3,b})$$

$$P_{6,b,i,(d)} = \partial_H(T_4 \| A_{2,d,b} \| K_{i,(b),(d)} \| B_{1,1-b})$$

where $b=0,1$, $d \in D$, $i=1,\ldots,4$ and (x) means either x or blank, as appropriate.

4.10. Claim: $\partial_H(T(M_p) \| P) = (\sum_{d \in D} 1^\circ_d \cdot 4^\circ_d \cdot 5^\circ \cdot 8^\circ)^\omega$.

4.10.1. Claim: $\partial_H(T(M_p) \| P) = \tau_{lp}(P_{1,0,1})$

Proof: entirely analogous to 4.6.1. □

4.10.2. Claim: $\tau_{lp}(P_{1,b,1}) = \sum_{d \in D} 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d})$

Proof: straightforward calculations. □

4.10.3. Claim: $P_{2,b,1,d} = P_{3,b,1,d}$

Proof: by definition. □

4.10.4. Claim: $\tau_{lp}(P_{3,b,1,d}) = \tau \cdot \tau_{lp}(P_{4,b,1,d})$

Proof: $P_{3,b,1,d} = 2^\circ_{d,b} \cdot P_{3,b,2,d}$

$P_{3,b,2,d} = ik \cdot P_{3,b,3,d} + jk \cdot P_{3,b,4,d} + kk \cdot P_{3,b,1,d}$

$P_{3,b,4,d} = 3^\circ_e \cdot P_{3,b,1,d}$

We find that $\{P_{3,b,i,d} \mid i=1,2,4\}$ is a cluster. We apply CFAR:

$$\tau_{lp}(P_{3,b,i,d}) = \tau \cdot \tau_{lp}(ik \cdot P_{3,b,3,d})$$
$$= \tau \cdot \tau_{lp}(ik \cdot 3^{\circ}_{d,b} \cdot P_{4,b,1,d}) = \tau \cdot \tau_{lp}(P_{4,b,1,d}) = \tau \cdot \tau_{lp}(P_{4,b,1,d}).$$

This proves the claim for i=1,2,4. It trivially holds for i=3. □

4.10.5.Claim: $\tau_{lp}(P_{4,b,1,d}) = \tau \cdot \sum_{i=1}^{4} (\tau\cdot)\, 4^{\circ}_{d} \cdot \tau_{lp}(P_{5,d,i,b})$.

Here $(\tau\cdot)$ is $\tau\cdot$ iff i=3,4.

Proof: $P_{4,b,1,d} = 4^{\circ}_{d} \cdot P_{5,b,1,d} + 2^{\circ}_{d,b} \cdot P_{4,b,2,d}$

$P_{4,b,2,d} = 4^{\circ}_{d} \cdot P_{5,b,2,d} + ik \cdot P_{4,b,3,d} + jk \cdot P_{4,b,4,d} + kk \cdot P_{4,b,1,d}$

Hence $P_{4,b,1,d}$ and $P_{4,b,2,d}$ form a cluster and (applying CFAR):

$\tau_{lp}(P_{4,b,1,d})$

$= \tau \cdot \tau_{lp}(4^{\circ}_{d} \cdot P_{5,b,1,d} + 4^{\circ}_{d} \cdot P_{5,b,2,d} + ik \cdot P_{4,b,3,d} + jk \cdot P_{4,b,4,d})$

$= \tau \cdot \tau_{lp}(4^{\circ}_{d} \cdot P_{5,b,1,d} + 4^{\circ}_{d} \cdot P_{5,b,2,d} + ik \cdot 4^{\circ}_{d} \cdot P_{5,b,3,d} + jk \cdot 4^{\circ}_{d} \cdot P_{5,b,4,d})$

$= \tau \cdot (4^{\circ}_{d} \cdot \tau_{lp}(P_{5,b,1,d}) + 4^{\circ}_{d} \cdot \tau_{lp}(P_{5,b,2,d}) + \tau \cdot 4^{\circ}_{d} \cdot \tau_{lp}(P_{5,b,3,d}) + \tau \cdot 4^{\circ}_{d} \cdot \tau_{lp}(P_{5,b,4,d}))$

$= \tau \cdot \sum_{i} (\tau\cdot)\, 4^{\circ}_{d} \cdot \tau_{lp}(P_{4,d,i,b})$. □

4.10.6.1.Claim: $\tau_{lp}(P_{5,b,i,d}) = \tau \cdot \sum_{j=1}^{4} (\tau\cdot)\, 5^{\circ}\cdot \tau_{lp}(P_{6,d,j,b})$ for i=1,2.

Again, $(\tau\cdot)$ is $\tau\cdot$ iff j=3,4.

Proof: $P_{5,b,1,d} = 5^{\circ}\cdot P_{6,b,1,d} + 2^{\circ}_{b,d} \cdot P_{5,b,2,d}$

$P_{5,b,2,d} = 5^{\circ}\cdot P_{6,b,2,d} + ik \cdot P_{5,b,3,d} + jk \cdot P_{5,b,4,d} + kk \cdot P_{5,b,1,d}$

Again $P_{5,b,1,d}$ and $P_{5,b,2,d}$ form a cluster and, by CFAR:

$\tau_{lp}(P_{5,b,i,d}) = \tau \cdot \tau_{lp}(5^{\circ}\cdot P_{6,b,1,d} + 5^{\circ}\cdot P_{6,b,2,d} + ik \cdot P_{5,b,3,d} + jk \cdot P_{5,b,4,d})$

$= \tau \cdot \sum_{j} (\tau\cdot)\, 5^{\circ}\cdot \tau_{lp}(P_{6,d,j,b})$, as before. □

4.10.6.2.Claim: $\tau_{lp}(P_{5,b,i,d}) = 5^{\circ}\cdot \tau_{lp}(P_{6,d,i,b})$ for i=3,4.

Proof: straightforward. □

4.10.7.Claim: $\tau_{lp}(P_{6,d,i,b}) = \tau \cdot \sum_{j=1}^{4} 8^{\circ}\cdot \tau_{lp}(P_{1,1-b,j,(d)})$.

Proof: $P_{6,d,1,b} = 8^\circ \cdot P_{1,1-b,1} + 2^\circ_{d,b} \cdot P_{6,b,2,d}$

$P_{6,d,2,b} = 8^\circ \cdot P_{1,1-b,2,d} + ik \cdot P_{6,b,3,d} + jk \cdot P_{6,b,4,d} + kk \cdot P_{6,b,1,d}$

$P_{6,d,3,b} = 8^\circ \cdot P_{1,1-b,3,d} + 3^\circ_{d,b} \cdot P_{6,b,1,d}$

$P_{6,d,4,b} = 8^\circ \cdot P_{1,1-b,4} + 3^\circ_{e} \cdot P_{6,b,1,d}$

We find that $\{P_{6,d,i,b} \mid i=1,\ldots,4\}$ is a cluster. Applying CFAR yields:

$$\tau_{lp}(P_{6,d,i,b}) = \tau \cdot \tau_{lp}(\Sigma_j 8^\circ \cdot P_{1,1-b,j,(d)}) = \tau \cdot \Sigma_j 8^\circ \cdot \tau_{lp}(P_{1,1-b,j,(d)}) \quad \square$$

4.10.8.1.Claim: $\tau_{lp}(P_{1,b,4}) = \tau \cdot \sum_{d\in D} 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d})$.

Proof: $P_{1,b,4} = \Sigma_d 1^\circ_d \cdot P_{2,b,4,d} + 3^\circ_e \cdot P_{1,b,1} = \Sigma_d 1^\circ_d \cdot 3^\circ_e \cdot P_{2,b,1,d} + 3^\circ_e \cdot \Sigma_d 1^\circ_d \cdot P_{2,b,1,d}$

Hence $\tau_{lp}(P_{1,b,4}) = \Sigma_d 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d}) + \tau \cdot \Sigma_d 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d})$

$= \tau \cdot \sum 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d})$. $\square$

4.10.8.2.Claim: $\tau_{lp}(P_{1,b,3,d}) = \tau \cdot \sum_{d\in D} 1^\circ_d \cdot \tau_{lp}(P_{2,b,1,d})$.

Proof: analogous to 4.10.8.1. $\square$

4.10.8.3.Claim: $\tau_{lp}(P_{1,b,2,d}) = \tau \cdot \sum_{d'\in D} 1^\circ_{d'} \cdot \tau_{lp}(P_{2,b,1,d'})$.

Proof:

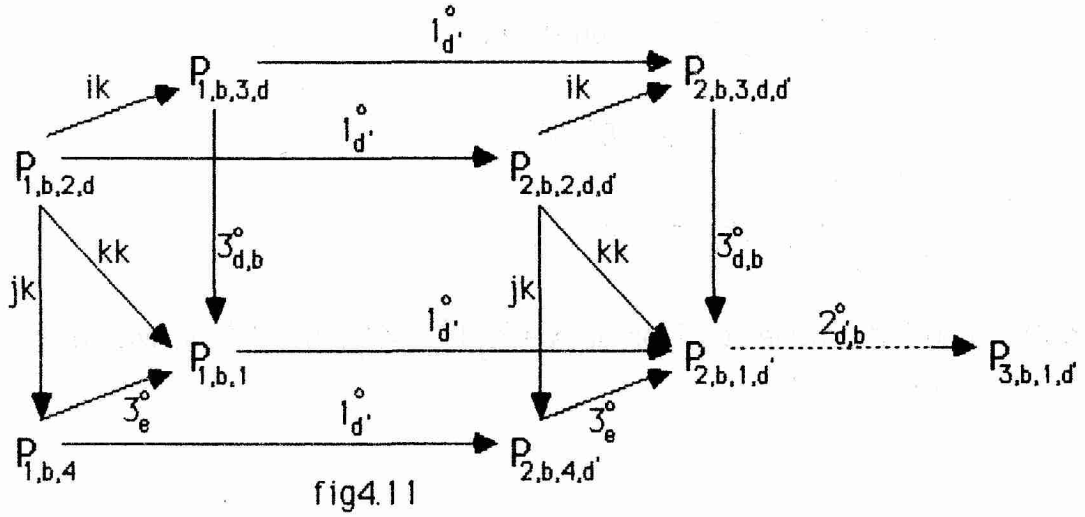


fig4.11

$P_{1,b,2,d} = \Sigma_{d'} 1^\circ_{d'} \cdot P_{2,b,2,d,d'} + ik \cdot P_{1,b,3,d} + jk \cdot P_{1,b,4} + kk \cdot P_{1,b,1}$

$= \Sigma_{d'} 1^\circ_{d'} \cdot (ik \cdot P_{2,b,3,d,d'} + jk \cdot P_{2,b,4,d'} + kk \cdot P_{2,b,1,d'})$

$$+ ik \cdot (\Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,3,d,d'} + 3^{\circ}_{b,d} \cdot P_{1,b,1})$$
$$+ jk \cdot (\Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,4,d'} + 3^{\circ}_{e} \cdot P_{1,b,1}) + kk \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,1,d'}$$
$$= \Sigma_{d'} 1^{\circ}_{d'} \cdot (ik \cdot 3^{\circ}_{d,b} \cdot P_{2,b,1,d'} + jk \cdot 3^{\circ}_{e} \cdot P_{2,b,1,d'} + kk \cdot P_{2,b,1,d'})$$
$$+ ik \cdot (\Sigma_{d'} 1^{\circ}_{d'} \cdot 3^{\circ}_{b,d} \cdot P_{2,b,1,d'} + 3^{\circ}_{b,d} \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,1,d'})$$
$$+ jk \cdot (\Sigma_{d'} 1^{\circ}_{d'} \cdot 3^{\circ}_{e} \cdot P_{2,b,1,d'} + 3^{\circ}_{e} \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,1,d'})$$
$$+ kk \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot P_{2,b,1,d'}$$

The claim follows. □

4.10.8.4. We can summarize 4.10.2. and 4.10.8.1. through 4.10.8.3:

$$\tau_{lp}(P_{1,b,i,d}) = (\tau\cdot)\, \Sigma_{d'} 1^{\circ}_{d'} \cdot \tau_{lp}(P_{2,b,1,d'}), \quad \text{where } (\tau\cdot) \text{ is blank iff } i=1.$$

4.10.9. Summing up 4.10.3 through 4.10.8 we find:

$$\begin{aligned}
\tau_{lp}(P_{2,b,1,d}) = \tau_{lp}(P_{3,b,1,d}) &= \tau \cdot \tau_{lp}(P_{4,b,d,1}) = \tau^2 \cdot \Sigma_i\, (\tau\cdot)\, 4^{\circ}_{d} \cdot \tau_{lp}(P_{5,b,i,d}) \\
&= \tau^2 \cdot \Sigma_i\, (\tau\cdot)\, 4^{\circ}_{d} \cdot \tau \cdot \Sigma_j\, (\tau\cdot)\, 5^{\circ} \cdot \tau_{lp}(P_{6,b,j,d}) \\
&= \tau^3 \cdot 4^{\circ}_{d} \cdot \tau \cdot \Sigma_j\, (\tau\cdot)\, 5^{\circ} \cdot \tau_{lp}(P_{6,b,j,d}) \\
&= \tau^3 \cdot 4^{\circ}_{d} \cdot \tau \cdot \Sigma_j\, (\tau\cdot)\, 5^{\circ} \cdot \tau \cdot \Sigma_k\, 8^{\circ} \cdot \tau_{lp}(P_{1,1-b,k,(d)}) \\
&= \tau^3 \cdot 4^{\circ}_{d} \cdot \tau^2 \cdot 5^{\circ} \cdot \tau \cdot \Sigma_k\, 8^{\circ} \cdot \tau_{lp}(P_{1,1-b,k,(d)}) \\
&= \tau^3 \cdot 4^{\circ}_{d} \cdot \tau^2 \cdot 5^{\circ} \cdot \tau \cdot \Sigma_k\, 8^{\circ} \cdot (\tau\cdot)\, \Sigma_{d'} 1^{\circ}_{d'} \cdot \tau_{lp}(P_{2,1-b,1,d'}) \\
&= \tau^3 \cdot 4^{\circ}_{d} \cdot \tau^2 \cdot 5^{\circ} \cdot \tau \cdot 8^{\circ} \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot \tau_{lp}(P_{2,1-b,1,d'}) \\
&= \tau \cdot 4^{\circ}_{d} \cdot 5^{\circ} \cdot 8^{\circ} \cdot \Sigma_{d'} 1^{\circ}_{d'} \cdot \tau_{lp}(P_{2,1-b,1,d'})
\end{aligned}$$

So if we denote $\sum_{d\in D} 1^{\circ}_{d} \cdot \tau_{lp}(P_{2,b,1,d})$ by P_b then

$$P_b = \sum_{d\in D} 1^{\circ}_{d} \cdot 4^{\circ}_{d} \cdot 5^{\circ} \cdot 8^{\circ} \cdot P_{1-b}$$

and $$P_{1-b} = \sum_{d\in D} 1^{\circ}_{d} \cdot 4^{\circ}_{d} \cdot 5^{\circ} \cdot 8^{\circ} \cdot P_b$$

So both equal the unique solution of $X = \Sigma_d 1^{\circ}_{d} \cdot 4^{\circ}_{d} \cdot 5^{\circ} \cdot 8^{\circ} \cdot X$, which is $(\Sigma_d 1^{\circ}_{d} \cdot 4^{\circ}_{d} \cdot 5^{\circ} \cdot 8^{\circ})^{\omega}$.

4.10.10. Finally we are in a position to prove claim 4.10:

$$\partial_H(T(M_p)\|P) = \tau_{If}(P_{1,0,1}) = \sum_{d\in D} 1^\circ_d \cdot \tau_{If}(P_{2,b,1,d}) = P_0 = (\sum_{d\in D} 1^\circ_d \cdot 4^\circ_d \cdot 5^\circ \cdot 8^\circ)^\omega. \ \square$$

4.11. In this subsection we will calculate $\partial_H(M_p\|M(Q))$.
As the notations from 4.4. and 4.8. collide, we have to rename some states. We choose to rename the states of M(Q) to N_1 and N_2. So:

$$M_p = M_1 = \sum_{d\in D} \underline{1}_d \cdot M_{2,d}$$

$$M_{2,d} = 4_d \cdot M_3 + \underline{8} \cdot \chi \qquad M(Q) = N_1 = \underline{5} \cdot N_2$$
$$M_3 = 5 \cdot M_4 + \underline{8} \cdot \chi \qquad N_2 = 8 \cdot N_1 + \underline{5} \cdot \chi$$
$$M_4 = \underline{8} \cdot M_1$$

$$\begin{aligned}\partial_H(M_p\|M(Q)) &= \partial_H(M_1\|N_1)\\ &= \textstyle\sum_d \underline{1}_d \cdot \partial_H(M_{2,d}\|N_1)\\ &= \textstyle\sum_d \underline{1}_d \cdot 4_d \cdot \partial_H(M_3\|N_1)\\ &= \textstyle\sum_d \underline{1}_d \cdot 4_d \cdot 5^\circ \cdot \partial_H(M_4\|N_2)\\ &= \textstyle\sum_d \underline{1}_d \cdot 4_d \cdot 5^\circ \cdot 8^\circ \cdot \partial_H(M_1\|N_1)\end{aligned}$$

So we find that $\partial_H(M_p\|M(Q)) = (\sum_d \underline{1}_d \cdot 4_d \cdot 5^\circ \cdot 8^\circ)^\omega$.

4.12. In the previous subsections we established $P \vdash M_p$ and $Q \vdash M(Q)$. Hence, by MAP, $P\|Q \vdash M_p\|M(Q)$. Consequently, $\partial_H(P\|Q) \vdash \partial_H(M_p\|M(Q))$, which was shown equal to $(\sum_d \underline{1}_d \cdot 4_d \cdot 5^\circ \cdot 8^\circ)^\omega$. The latter being robust, we may apply ROP and find $\tau\cdot\partial_H(P\|Q)\cdot\delta = \tau\cdot(\sum_d \underline{1}_d \cdot 4_d \cdot 5^\circ \cdot 8^\circ)^\omega \cdot \delta$. Since these processes are perpetual, this implies $\tau_I\partial_H(P\|Q) = (\tau\cdot)(\sum_d \underline{1}_d \cdot 4_d)^\omega$. This proves that CABP is a correct communication protocol.

5. References.

[1] BAETEN, J.C.M., J.A. BERGSTRA & J.W.KLOP, *Conditional axioms and α/β calculus in process algebra*, CWI Report CS-R8502, 1985.

[2] BERGSTRA, J.A. & J.W. KLOP, *Process Algebra for Synchronous Communication*, Information & Control 60 (1/3), pp. 109-137, 1984.

[3] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes with Abstraction*, Theoretical Computer Science 37(1), pp 77-121, 1985.

[4] BERGSTRA, J.A. & J.W. KLOP, *Verification of an Alternating Bit Protocol by means of Process Algebra*, CWI Report CS-R8404, 1984.

[5] BERGSTRA, J.A. & J.W. KLOP, *Fair FIFO queues satisfy an algebraic criterion for protocol correctness*, CWI Report CS-R8405, 1984.

[6] BERGSTRA, J.A. & J.W. KLOP, *A complete inference system for regular processes with silent moves*, CWI Report CS-R8420, 1984.

[7] MILNER, R., *A Calculus of Communicating Systems*, Lecture Notes in Computer Science No 92, Springer Verlag, New York/Berlin, 1980.

[8] PARROW, J., *Fairness properties in Process Algebra with applications in Comminucation Protocol Verification*, PhD thesis, Uppsala University, Uppsala, 1985.

[9] SCHOONE, A.A. & J. VAN LEEUWEN, *Verification of balanced link-level Protocols*, State University of Utrecht preprint RUU-CS-85-12, Utrecht 1985.

[10] TANENBAUM, A.S., *Computer Networks*, Prentice Hall, 1981.

[11] VAANDRAGER, F.W., *Verification of two communication protocols by means of Process algebra*, CWI Report CS-R8608, 1986.