J.A. BERGSTRA & J.W. KLOP

AN ABSTRACTION MECHANISM FOR PROCESS ALGEBRAS

Preprint

1980 Mathematics subject classification: 68B10, 68C01, 68D25, 68F20

1982 CR. Categories: D.1.3, F.1.1, F.1.2, F.3.2

An abstraction mechanism for process algebras [*]

by

J.A. Bergstra & J.W. Klop

ABSTRACT

The problem of hiding internal steps in a system of processes is addressed within the framework of process algebra. Using a graph representation of processes that represents process merge as a cartesian product on graphs, we introduce bisimulation modulo internal steps, an equivalence relation on the algebra of processes. Bisimulation is not a congruence, but we identify an important subalgebra, the algebra of guarded and bounded processes, on which bisimulation is a congruence.

This congruence is shown to be a useful abstraction tool in two examples concerning simple communication protocols.

---

[*] This report will be submitted for publication elsewhere.

INTRODUCTION

Let A be a finite domain of atomic actions. We conceive a process as some finite or infinite configuration of these atomic actions; here one may think of rooted transition graphs, labeled by atomic actions, which include trees and sequences.

In the discussion at the end of the paper we will expand on different points of view one may have concerning the very concept of a process. We will specify the behaviour of processes by means of algebraic equation systems. The main operators on processes that we consider are these:

$+$ *choice*
$\cdot$ *sequential composition*
$\parallel\!\!\!\perp$ *left merge*
$\parallel$ *merge*

and the algebraic laws specifying their behaviour constitute the following axiom system PA:

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y).z = x.z + y.z$	A4
$(x.y).z = x.(y.z)$	A5
$x \parallel y = x \parallel\!\!\!\perp y + y \parallel\!\!\!\perp x$	M1
$a \parallel\!\!\!\perp x = a.x$	M2
$ax \parallel\!\!\!\perp y = a(x \parallel y)$	M3
$(x + y) \parallel\!\!\!\perp z = x \parallel\!\!\!\perp z + y \parallel\!\!\!\perp z$	M4

In Section 1 we comment on these axioms.

Though not explicit in PA, it may be the case that different atoms have a different status. Especially one may wish to call certain actions $E \subseteq A$

visible (external) and the other actions invisible (internal). The typical way for internal actions to arise is as internal communications in a network of processes; this will be explained in detail in Section 4 and illustrated by the two examples in Sections 5 and 6.

Now the important problem (which has been addressed in the litterature previously at several occasions, see the discussion in Section 7) is to provide an abstraction mechanism to hide invisible actions. Such a mechanism is explained in Section 3, and constitutes the main result of this paper.

The structure of this paper is as follows:

1. SEMANTICS OF PA

1.1. Process algebras.

In the axioms presented in the Introduction, 'a' ranges over atomic actions and x,y,z range over arbitrary processes. Instead of x.y we also write xy.

A *process algebra* over A is an algebra

$$(P, +, \cdot, \mathbin{\rlap{\rule[0pt]{0.3em}{0.4pt}}\rule[-0.2em]{0.4pt}{0.8em}}\, , \|\, , \{a\}_{a \in A})$$

satisfying the axioms A1-5, M1-3 where M2 and M3 should be read as axiom schemes in which all constants $a \in A$ have to be substituted.

We will now describe the most important model of PA.

1.2. A_ω , the initial algebra of PA over A.

Let $T_A(+, \cdot, \mathbin{\rlap{\rule[0pt]{0.3em}{0.4pt}}\rule[-0.2em]{0.4pt}{0.8em}}\, , \|)$ be the free algebra of closed $(+, \cdot, \mathbin{\rlap{\rule[0pt]{0.3em}{0.4pt}}\rule[-0.2em]{0.4pt}{0.8em}}\, , \|)$-terms over A. PA generates a congruence $\equiv_{PA}$ on this term algebra. Now A_ω is just:

$$T_A(+, \cdot, \mathbin{\rlap{\rule[0pt]{0.3em}{0.4pt}}\rule[-0.2em]{0.4pt}{0.8em}}\, , \|) \,/\, \equiv_{PA} \,.$$

There are some important observations about A_ω to be made. Let $T_A(+, \cdot)$ be the free algebra of $(+, \cdot)$-terms over A. Then the natural homomorphism ϕ:

$$\phi : T_A(+, \cdot) \longrightarrow A_\omega$$

is a surjection on A_ω. This means that each $p \in A_\omega$ can be represented as a term not containing $\mathbin{\rlap{\rule[0pt]{0.3em}{0.4pt}}\rule[-0.2em]{0.4pt}{0.8em}}$ and $\|$.

Secondly, two terms $t_1, t_2 \in T_A(+, \cdot)$ have identical meaning in A_ω if and only if they can be proved equal using A1-5 only. (In data type terminology: A_ω is an enrichment of $T_A(+, \cdot) \,/\, \equiv_{A1-5} \,.$)

The main drawback of A_ω is that it does not allow solving fixed point equations: in A_ω there is no x with x = ax + b. Therefore other models must be taken into account, because fixed point equations are the main specification tool of process algebra.

1.3. A_n: the initial model modulo n.

We will now develop a family of finite models of PA, all of which are homomorphic images of A_ω.

Let $(x)_n$, the n-th projection of $x \in A_\omega$, be defined for $n \in \{1, 2, ..\}$ as

4

follows:

$$(a)_n = a$$

$$(ax)_1 = a$$

$$(ax)_{n+1} = a(x)_n$$

$$(x + y)_n = (x)_n + (y)_n.$$

The domain of A_n consists of all $(x)_n$ for $x \in A$ (i.e. those y for which $y = (y)_n$).

Now $+^n, \cdot^n, \underline{\|}^n, \|^n$ are defined on A_n as follows:

$$x +^n y = (x + y)_n$$

$$x \cdot^n y = (x \cdot y)_n$$

$$x \underline{\|}^n y = (x \underline{\|} y)_n$$

$$x \|^n y = (x \| y)_n.$$

This gives us

$$A_n = (A_\omega)_n (+^n, \cdot^n, \underline{\|}^n, \|^n),$$

which we will write for simplicity as

$$A_n = A_n (+, \cdot, \underline{\|}, \|).$$

There are obvious homomorphisms $\phi_n : A_\omega \longrightarrow A_n$, and for each n there is a homomorphism $\phi_n^{n+1} : A_{n+1} \longrightarrow A_n$ such that the following diagram commutes:

REMARK. It was shown in [5] that in A_n each equation of the form $x = t(x)$ has a solution. If x occurs guarded in t, like in $t(x) = (ax \| a(x+b)) + cx$, then the unique solution is found by iterated (n times) substitution. If x is not guarded in t, like in $t(x) = x \| (ax+b) + cx$, then a more combinatorial argument shows the existence of a solution.

1.4. The projective limit A^∞ of the structures A_n.

The ingredients described in the previous subsection, viz.

$$\left\{ \begin{array}{l} A_n \\ \phi_n^{n+1} : A_{n+1} \longrightarrow A_n \\ \phi_n : A_\omega \longrightarrow A_n \end{array} \right.$$

together with the commuting diagram above lead to a *projective limit* A^∞.
Elements of A^∞ are the projective sequences, i.e. sequences $(p_1, p_2, \ldots)$ with
$p_n \in A_n$ and $p_n = \phi_n^{n+1}(p_{n+1})$ for all n. The operations $+, \cdot, \lfloor\!\lfloor$ and $\|$ are defined component-wise.

A^∞ serves as a *standard model* of PA in our considerations. It is in fact
an algebraic reconstruction of the space of uniform processes over A as defined in DE BAKKER & ZUCKER [3,4]. The model A^∞ inherits its ability to solve
fixed point equations from the structures A_n.

1.5. $B^\infty(E,I)$: processes with bounded internal traces.

We assume in this subsection that $A = E \cup I$ and $E \cap I = \emptyset$. Actions in E are external, and actions in I are internal.

We focus attention on a subalgebra $B^\infty(E,I)$ of A^∞ $(= (E \cup I)^\infty)$. This subalgebra $B^\infty(E,I)$ contains those processes $(p_1, p_2, \ldots)$ in $(E \cup I)^\infty$ which satisfy the following condition:

for each $k \geqslant 1$ there is a bound $\ell \geqslant 1$ on the length of traces consisting of internal steps only that occur in the p_n and which start at level k.
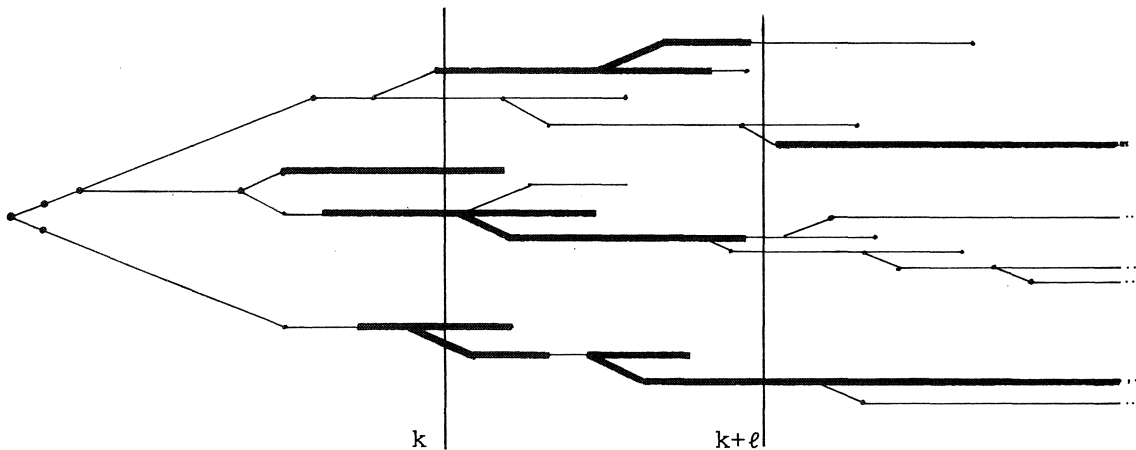
(Heavy lines denote internal steps.)

This rather informal description is clarified by the following formal definition.

The operation $tr: A_\omega \rightarrow A^*$ defines the trace set of $p \in A_\omega$ as follows:

$$tr(a) = \{a\}$$
$$tr(ax) = a * tr(x)$$
$$tr(x + y) = tr(x) \cup tr(y).$$

If $\alpha \in A^*$, then $[\alpha]_n$ denotes its n-th component (provided $n \leqslant lth(\alpha)$).
Now the condition on $B^\infty(E,I)$ reads as follows:

$p = (p_1, p_2, \ldots)$ has *bounded internal traces* if for each k there is an $\ell \geqslant 1$ such that for all $\alpha \in tr(p_{k+\ell})$:

$$[\alpha]_k \in I \implies (lth(\alpha) \leqslant k+\ell \vee \exists\, 1 \leqslant j \leqslant \ell\ [\alpha]_{k+j} \in E).$$

$B^\infty(E,I)$ contains all processes of $(E \cup I)^\infty$ which have bounded internal traces. It is not difficult to see that $B^\infty(E,I)$ is indeed a *subalgebra* of $(E \cup I)^\infty$.

The importance of $B^\infty(E,I)$ is that it provides a process algebra where in principle abstraction from internal steps is conceivable. We can hardly expect for instance to eliminate internal steps from a process like $a + i^\omega$ ($i^\omega = i.i^\omega$) which diverges when choosing for the internal option.

1.5.1. THEOREM. *Let* $x = t(x)$ *be an equation in which each occurrence of* x *is guarded (preceded in the formation tree of* t*) by some external action.*

Then the unique fixed point of $x = t(x)$ *in* A^∞ *is contained in* $B^\infty(E,I)$.

PROOF. In this proof we work on terms modulo the equality generated by A2,A5 (associativity of • and +). Consider A1,A4,M1-4 and use these axioms as rewrite rules from left to right. Then each term t can be rewritten to a normal form t'. (See [6] for an analysis of the term rewriting system associated with PA.)

During this rewriting procedure, the symbols (actions) from t can be traced to their positions in t' (for this reason we exclude A3 for the time being). In other words, each symbol $\in E \cup I$, or rather symbol occurrence, in t' derives from a unique 'ancestor' symbol (occurrence) in t.

Some terminology: the inverse notion of 'ancestor' is 'descendant'. Furthermore, if t is a term and u_1, u_2 are two symbol occurrences in t, write $u_1 \succ u_2$ ("u_1 guards u_2") if u_1 is above u_2 in the formation tree of t.

Now let $t \rightarrow \ldots \rightarrow t'$, where ' $\rightarrow$ ' denotes a rewrite step, and observe:

(1) If u is a symbol occurrence in t, then the descendants of u in t' are incomparable w.r.t. $\succeq$.

(2) If u_1, u_2 are symbol occurrences in t and $u_1 \succeq u_2$, then every descendant of u_2 in t' is guarded by some descendant of u_1 in t'.

From these observations it is straightforward to derive the fact that the length of internal traces starting at level n in the fixed point of $x = t(x)$, cannot exceed the bound p.n where p is the number of occurrences of internal actions in t(x). $\square$

2. GRAPH REPRESENTATION OF FINITE PROCESSES

2.1. Process graphs.

A *process graph* over a set of atoms A is a rooted, directed multigraph whose edges are labeled by elements from A.

Example:
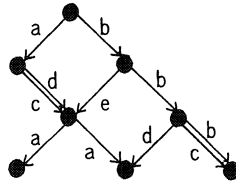
In this paper only *acyclic* and *finite* process graphs will be used.

Let $\mathcal{G}_A$ be the collection of acyclic and finite process graphs. Let ϕ denote the graph without edges. A surjective mapping $[\![\ ]\!]: \mathcal{G}_A \backslash \{\phi\} \rightarrow A_\omega$ is defined as follows:

$$[\![\ \bullet \xrightarrow{\ a\ } \ ]\!] = a;$$

if $\alpha_1, \ldots, \alpha_k$ are the edges starting from the root of g and a(i) are the labels, p(i) the endpoints of these edges, then

$$[\![g]\!] = \sum_{i=1}^{k} a(i) \cdot [\![g \lceil p(i)]\!].$$

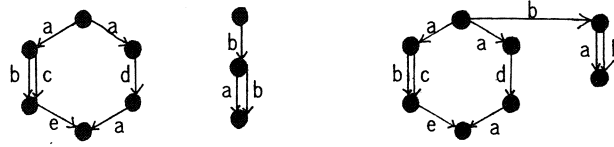Here $g \lceil p(i)$ is the subgraph of g with root p(i) and $a(i)[\![\phi]\!]$ is just a(i). In the case of the example above one obtains:

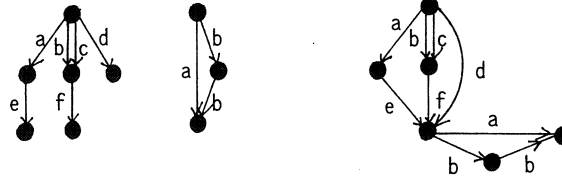$$a(ca + da) + b(ea + b(d + c + b)).$$

2.2. Operations on process graphs.

$\mathcal{G}_A$ will now be enriched with four operations: $+, \cdot, \times\!\!\perp$, and $\times$.

Let $g_1, g_2 \in \mathcal{G}_A$. Then $g_1 + g_2$ is obtained by glueing the roots of g_1 and g_2 together. Example:

Further, $g_1 \cdot g_2$ is obtained by glueing together the root of g_2 and *all* endpoints of g_1, as in:

Next, $g = g_1 \times g_2$ is obtained by taking g to be the cartesian product of g_1 and g_2, as follows. Let $r_0, \ldots, r_n$ be the nodes of g_1 and $s_0, \ldots, s_m$ the nodes of g_2; r_0 and s_0 are the respective roots. Then the product graph g has nodes (r_i, s_j) where $i \leqslant n$, $j \leqslant m$, the root is (r_0, s_0) and the edges are given by:

if $r_i \xrightarrow{\ a\ } r_{i'}$ is in g_1, then $(r_i, s_j) \xrightarrow{\ a\ } (r_{i'}, s_j)$ is in g;

if $s_j \xrightarrow{\ a\ } s_{j'}$ is in g_2, then $(r_i, s_j) \xrightarrow{\ a\ } (r_i, s_{j'})$ is in g.

Note that each edge in g_1 occurs $|g_2| = m+1$ times in g. Thus if $e(g_i)$ is the number of edges in g_i and $|g_i|$ is the number of nodes in g_i, then

$$|g| = |g_1| \cdot |g_2|, \text{ and}$$

$$e(g) = e(g_1) \cdot |g_2| + e(g_2) \cdot |g_1|.$$
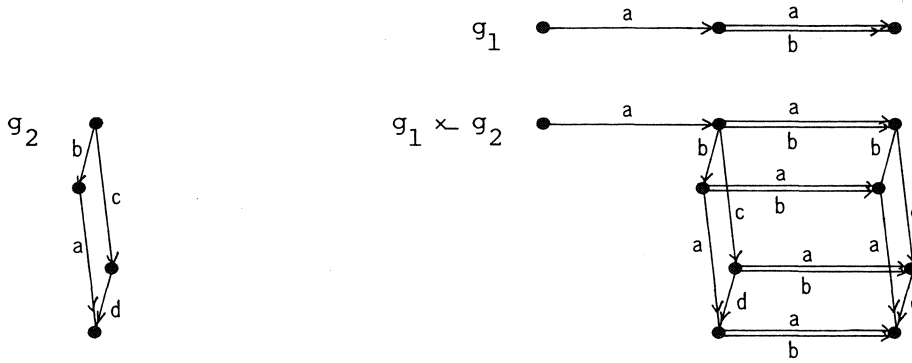
Example:
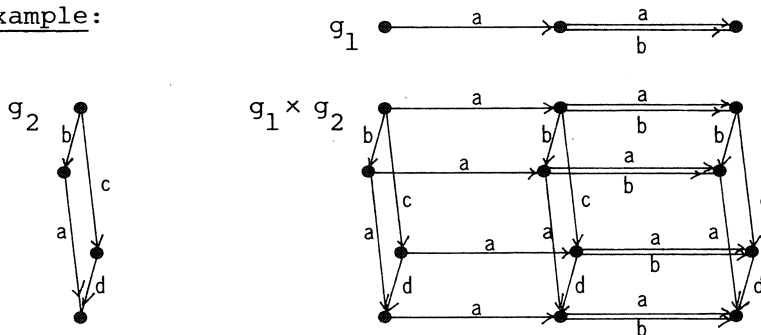
Finally, if $g_1 \neq \emptyset$ then $g_1 \times\!\!\!- g_2$ is defined just as $g_1 \times g_2$ but now dele-
ting all edges of the form

$$(r_0, s_0) \longrightarrow (r_0, s_i),$$

and thereafter removing all inaccessible nodes (as well as the edges they
support).

If $g_1 = \emptyset$ then $g_1 \times\!\!\!- g_2 = \emptyset$.

Example:

2.3. Mapping graphs onto processes.

Let $\mathcal{G}_A^- = \mathcal{G}_A - \{\emptyset\}$. Now $\mathcal{G}_A^-$ can be made into an algebra of the same signature
of process algebras:

$$\mathcal{G}_A^- (+, \cdot, \times\!\!\!-, \times).$$

This algebra is not a process algebra - it does not satisfy $(x + y).z = x.z + y.z$
for instance. However, there is the following theorem, which says that
$\mathcal{G}_A^- (+, \cdot, \times\!\!\!-, \times)$ is a representation of A_ω.

THEOREM. $[\![\ ]\!]: \mathcal{G}_A^- (+, \cdot, \times\!\!\!-, \times) \longrightarrow A_\omega (+, \cdot, |\!\!\lfloor, |\!|)$ *is a homomorphism.*

PROOF. $[\![g_1 + g_2]\!] = [\![g_1]\!] + [\![g_2]\!]$ and $[\![g_1 \cdot g_2]\!] = [\![g_1]\!].[\![g_2]\!]$ is immediate.

That $[\![g_1 \times\!\!\!- g_2]\!] = [\![g_1]\!] |\!\!\lfloor [\![g_2]\!]$ and $[\![g_1 \times g_2]\!] = [\![g_1]\!] |\!| [\![g_2]\!]$ is shown with a straight-
forward induction on $|g_1| + |g_2|$. $\square$

3. SEMANTICS OF PA IN THE PRESENCE OF INTERNAL ACTIONS

In this section we assume that the alphabet A of atomic actions is partitioned into two disjoint and nonempty subsets E and I. Actions in E are considered external, or visible. Actions from I are internal (invisible, inert). As a rule $a,b,c,d,\ldots$ will denote external actions, and $i,j,i',j',\ldots$ will denote internal actions.

Our aim is to find an equivalence relation which identifies processes having the same behaviour as seen from outside. To provide some intuition about these matters, we consider two small examples first. Let $p = aib$, $q = ab$:

```
p:   a|       q:   a|
     i|            b|
     b|
```

Then we postulate that p and q have equivalent external behaviour. However, let now $p' = a(ib + jc)$ and $q' = a(b + c)$.

```
p':   a          q':   a
     i j               b c
    b   c
```
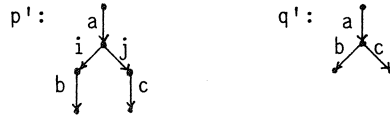
In this case we view p' and q' as inequivalent. The argument is that there is a context in which p' deadlocks but q' does not. To see this note that $b + c$ represents a guarded command. In CSP-like notation we could interpret p' and q' as follows:

$$p' = C_1!x \ ((x:=x; \ C_1!y) \ \Box \ (y:=y; \ C_2?y))$$

$$q' = C_1!x \ (C_1!y \Box C_2?y)$$

Let $r = C_1?z; \ C_2!z$, then

$$[q'\|r] = (z:=x);(y:=z), \text{ and}$$

$$[p'\|r] = (z:=x);(x:=x; \ \text{deadlock} \Box y:=y; \ y:=z).$$

This finishes our small examples, now we proceed with a formal description of bisimulation modulo I.

3.1. Bisimulation modulo internal actions.

A *trace* σ is a possibly empty finite string over $E \cup I$ (thus $\sigma \in (E \cup I)*$).
With $e(\sigma)$ we denote the trace σ in which all internal steps are erased (in
detail: e is the homomorphism e: $(E \cup I)* \longrightarrow E*$ generated by $e(\Lambda) = \Lambda$,
$e(i) = \Lambda$, $e(a) = a$, where $i \in I$, $a \in E$ and Λ is the empty word).

Consider a process graph g over $E \cup I$. A *path* $\pi: s_0 \longrightarrow\!\!\!\!\!\gg s_k$ in g is a
sequence of the form

$$s_0 \xrightarrow[h_0]{\ell_0} s_1 \xrightarrow[h_1]{\ell_1} \cdots \xrightarrow[h_{k-1}]{\ell_{k-1}} s_k$$

$(k \geqslant 0)$ where the s_i are nodes, the h_i are edges between s_i and s_{i+1}, and
each ℓ_i is the label of edge h_i. (The h_i are needed because we work with
multigraphs.) The trace $\textit{trace}(\pi)$ associated to this path is just $\ell_0\ell_1\ldots\ell_{k-1}$.

3.1.1. DEFINITION. A *bisimulation modulo* I between two process graphs g_1 and
g_2 is a relation R on NODES$(g_1) \times$ NODES(g_2) satisfying the following conditions:

(i) $(\text{ROOT}(g_1), \text{ROOT}(g_2)) \in R$,

(ii)   Domain$(R) = $ NODES(g_1) and Codomain$(R) = $ NODES(g_2),

(iii)   For each pair $(s_1, s_2) \in R$ and for each path $\pi_1: s_1 \longrightarrow\!\!\!\!\!\gg t_1$ in g_1 there
        is a path $\pi_2: s_2 \longrightarrow\!\!\!\!\!\gg t_2$ in g_2 such that $(t_1, t_2) \in R$ and
        $e(\textit{trace}(\pi_1)) = e(\textit{trace}(\pi_2))$.
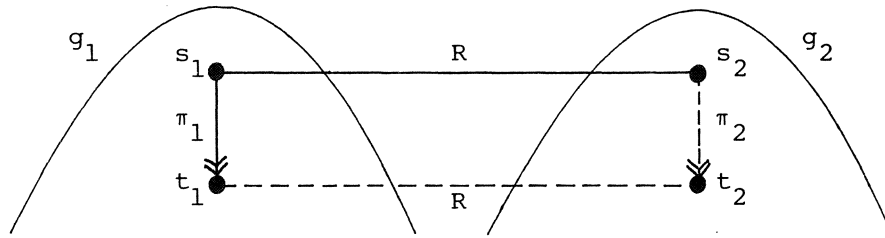
(iv)   Likewise for each pair $(s_1, s_2) \in R$ and for each path $\pi_2: s_2 \longrightarrow\!\!\!\!\!\gg t_2$
        in g_2 there is a path $\pi_1: s_1 \longrightarrow\!\!\!\!\!\gg t_1$ in g_1 such that $(t_1, t_2) \in R$
   and $e(\textit{trace}(\pi_1)) = e(\textit{trace}(\pi_2))$.
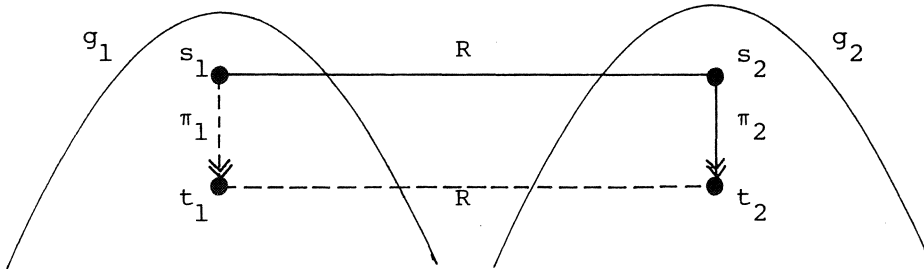
3.1.2. <u>DEFINITION</u>. Process graphs g_1 and g_2 are *bisimulation equivalent (modulo I)* if there is a bisimulation (modulo I) R between g_1 and g_2.

3.1.3. <u>NOTATION</u>. We write $g_1 \underset{I}{\rightleftharpoons} g_2$ for bisimulation equivalence modulo I. Usually we will omit the subscript I.

It is easy to see that $\rightleftharpoons$ is indeed an equivalence relation.

3.1.4. <u>EXAMPLES</u>. In this subsection we present several positive and negative examples of bisimulation modulo I.

<u>Positive examples</u>:
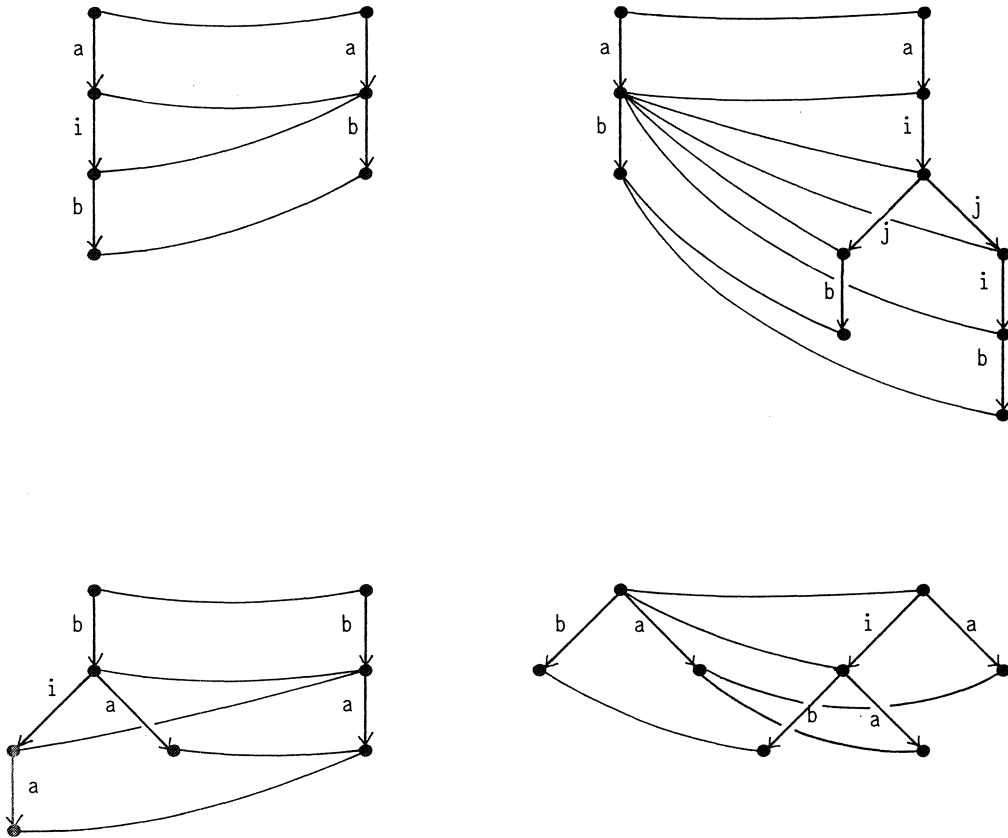
Negative examples. In the pictures below we see the pairs which any potential bisimulation of the graphs must contain. However (consider the first picture below) from the node s_1 a c-step to s_1' is possible and not from s_2, from which it follows that no proper bisimulation can be found. The difficulty reveals itself in the picture by the pair of heavy lines.
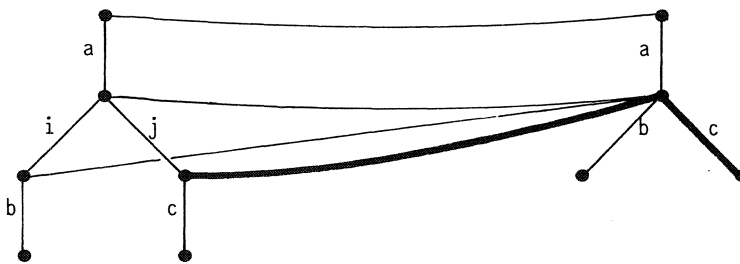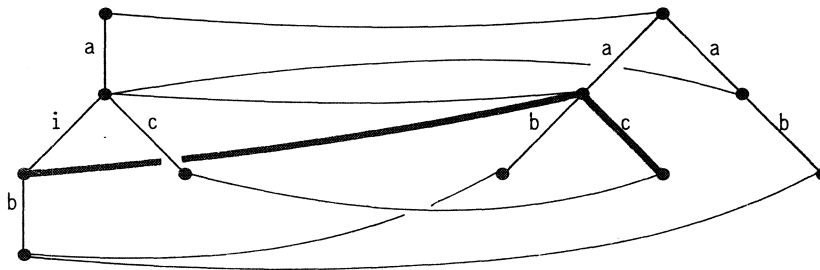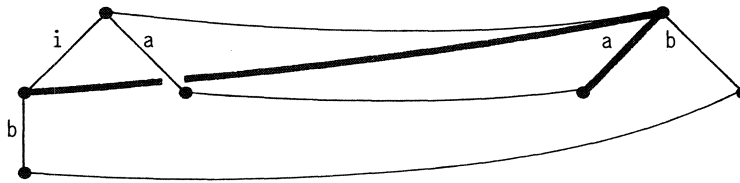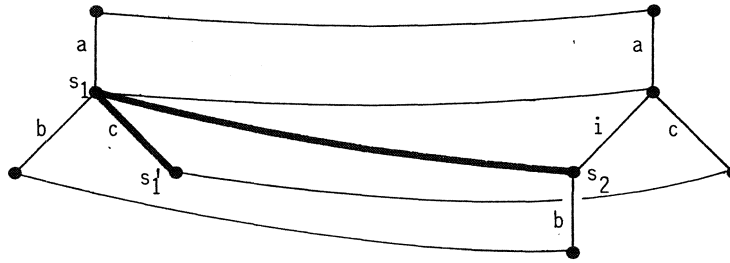
The last example was also discussed in the introduction of this section.

3.2. Bisimulation is (conditionally) preserved by the operators on process graphs.

Let a process graph be called *externally guarded* if it does not allow a non-empty internal first step. We will in short call such graphs *guarded*.

E.g. is not guarded. A guarded process graph is called in MILNER [13] 'stable'.

__THEOREM__. *Suppose $g_1 \leftrightarrows g_1'$ and $g_2 \leftrightarrows g_2'$, then:*

(i) $g_1 \cdot g_2 \leftrightarrows g_1' \cdot g_2'$

(ii) $g_1 \times g_2 \leftrightarrows g_1' \times g_2'$

(iii) $g_1 + g_2 \leftrightarrows g_1' + g_2'$ *provided g_1, g_2, g_1', g_2' are guarded,*

(iv) $g_1 \times\!\!\!\_ \; g_2 \leftrightarrows g_1' \times\!\!\!\_ \; g_2'$ *provided g_1 and g_1' are guarded.*

__PROOF__. (i) is illustrated by the picture below:
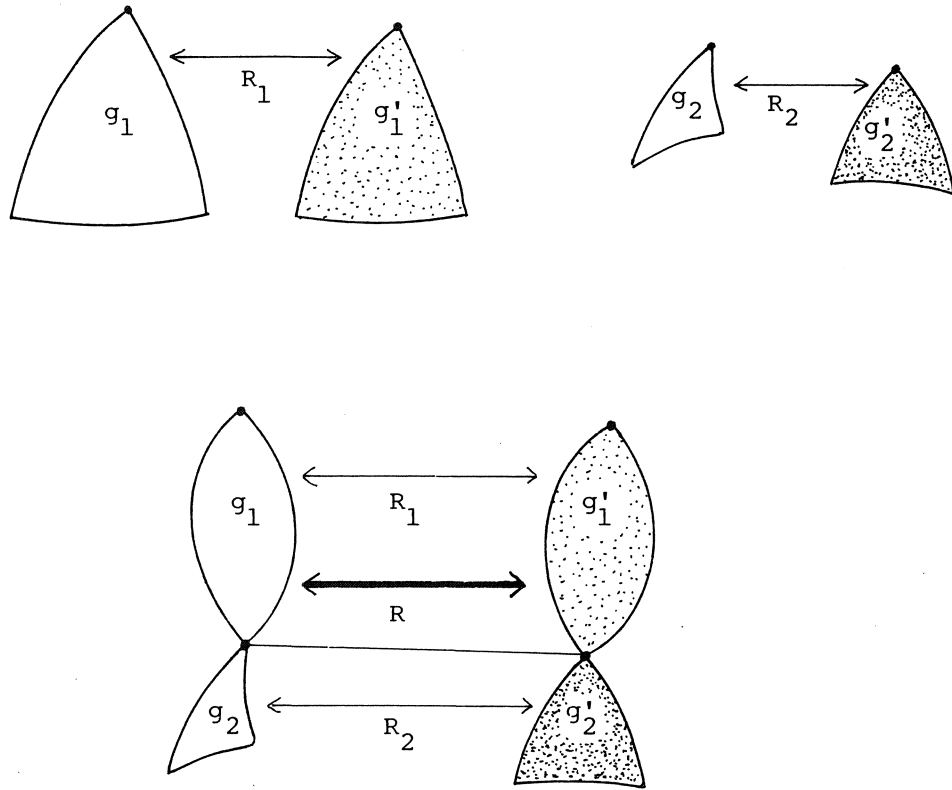
Here R is just the image of $R_1 \cup R_2$ after the appropriate identifications have been made. (Viz. identifying all endpoints of g_1 with the root of g_2 and like-wise for g_1' and g_2'.)

(ii) Suppose $g_1 \xleftrightarrow[R_1]{} g_1'$ and $g_2 \xleftrightarrow[R_2]{} g_2'$, then $g_1 \times g_2 \xleftrightarrow[R_1 \otimes R_2]{} g_1' \times g_2'$

where $R_1 \otimes R_2$ contains $((s_1,s_2),(s_1',s_2'))$ iff $(s_1,s_1') \in R_1$ and $(s_2,s_2') \in R_2$. It is easily verified that $R_1 \otimes R_2$ is a bisimulation indeed.

(iii) Again a picture is useful.

Thus $g_1 + g_2 \xLeftrightarrow{R_1 \cup R_2} g_1' + g_2'$. The requirement that the graphs are guarded, guarantees that R_1 and R_2 do not unpleasantly interfere after identifying the roots of g_1,g_2 and g_1',g_2'. (Cf. the second negative example in 3.1.4.)
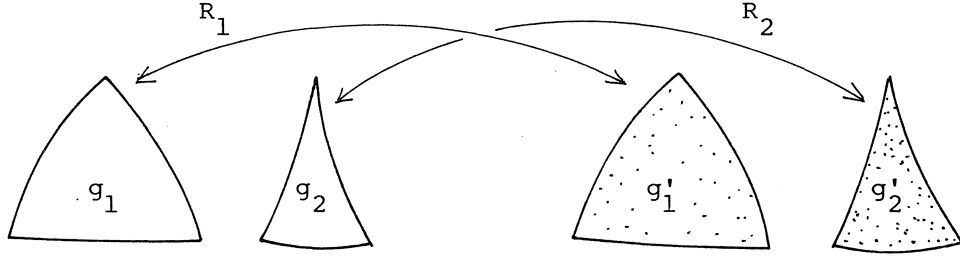
(iv) As defined in 2.2 $g_1 \underset{\textstyle\;}{\times} g_2$ is just a subgraph of $g_1 \times g_2$. Taking the appropriate subset of $R_1 \otimes R_2$ we obtain a correct bisimulation.

The proviso that g_1,g_1' are guarded is essential; e.g. let g_i,g_i' be as follows:

then $g_1 \underset{\textstyle\;}{\times} g_2$ is

and $g_1' \underset{\textstyle\;}{\times} g_2'$ is
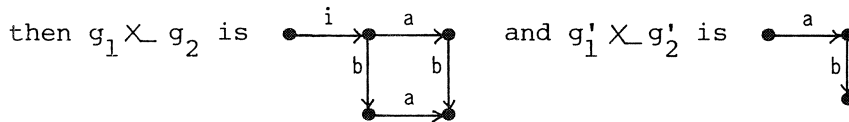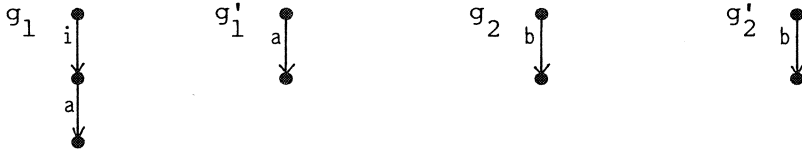
and these graphs are not bisimulation equivalent. (In algebraic terms:

$$ia \;\underline{\|}\; b = i(a\|b) = i(ab + ba), \quad a\,\underline{\|}\,b = ab.)$$

3.3. Bisimulation equivalence modulo I on $(E \cup I)_\omega$.

The notion of bisimulation (modulo I) on graphs immediately leads to a corresponding notion on finite processes.

DEFINITION. For $p,q \in (E \cup I)_\omega$ we define $p \xLeftrightarrow{} q$ if there are process graphs g_p and g_q over $E \cup I$ such that $p = [\![g_p]\!]$, $q = [\![g_q]\!]$ and $g_p \xLeftrightarrow{} g_q$.

In order to verify that bisimulation equivalence (mod.I) has been correctly defined this way, we must check that $[\![g_1]\!] = [\![g_2]\!]$ implies $g_1 \underset{\longleftrightarrow}{} g_2$. A formal proof of this fact requires an essentially straightforward but nevertheless slightly tedious induction on the number of nodes of g_1 and g_2.

3.4. $\underset{I}{\longleftrightarrow}$ on guarded processes: $G_\omega(E,I)/\underset{I}{\longleftrightarrow}$.

DEFINITION. $G_\omega(E,I)$ is the subalgebra of $(E \cup I)_\omega$ consisting of (externally) guarded processes only.

THEOREM. $\underset{I}{\longleftrightarrow}$ *is a congruence relation on* $G_\omega(E,I)$.

PROOF. This follows immediately from the analysis of $\underset{}{\longleftrightarrow}$ on process graphs (Theorem 3.2). $\square$

Notice that E_ω is a subalgebra of $G_\omega(E,I)$.

PROPOSITION. *Restricted to* E_ω, $\underset{I}{\longleftrightarrow}$ *coincides with identity (in* $(E \cup I)_\omega$).

PROOF. One must show that whenever g_p and g_q are process graphs over E with $g_p \underset{I}{\longleftrightarrow} g_q$ we have in $(E \cup I)_\omega$ that $[\![g_p]\!] = [\![g_q]\!]$. This is done using inductions on $|\text{NODES}(g_p)|$ and $|\text{NODES}(g_q)|$. $\square$

3.5. External projections of finite processes.

In the presence of internal steps the original projections $(x)_n$ loose much of their attraction. For instance,

$$\text{iabc} \underset{I}{\longleftrightarrow} \text{abc}$$

but

$$(\text{iabc})_2 \underset{I}{\not\longleftrightarrow} (\text{abc})_2.$$

This leads to introducing *external* projections that do not suffer from this kind of anomaly w.r.t. $\underset{I}{\longleftrightarrow}$. For $x \in (E \cup I)_\omega$ and $n \geqslant 1$ we define $(x)_n^e$ as follows:

$$(i)_n^e = i \ (i \in I); \qquad (a)_n^e = a \ (a \in E);$$

$$(ix)_n^e = i(x)_n^e; \qquad (ax)_1^e = a;$$

$$(ax)_{n+1}^e = a(x)_n^e; \qquad (x + y)_n^e = (x)_n^e + (y)_n^e.$$

3.5.1. <u>THEOREM</u>. *For* $p,q \in (E \cup I)_{\omega}$, $n \geq 1$:

$$p \underset{I}{\rightleftharpoons} q \implies (p)_n^e \underset{I}{\rightleftharpoons} (q)_n^e.$$

<u>PROOF</u>. Obvious. $\square$

3.5.2. <u>THEOREM</u>. *For* $p,q \in (E \cup I)_{\omega}$,

$$p \underset{I}{\rightleftharpoons} q \iff \forall n \geq 1 \quad (p)_n^e \underset{I}{\rightleftharpoons} (q)_n^e.$$

<u>PROOF</u>. For n sufficiently large we have $(p)_n^e = p$ and $(q)_n^e = q$. $\square$

3.5.3. <u>LEMMA</u>. *The following properties hold for* $(.)_n^e$ *on* $(E \cup I)_{\omega}$:

(i) $(x \cdot y)_n^e = ((x)_n^e \cdot (y)_n^e)_n^e$

(ii) $(x \| y)_n^e = ((x)_n^e \| (y)_n^e)_n^e$

(iii) $(x \underset{}{\lfloor\!\lfloor} y)_n^e = ((x)_n^e \underset{}{\lfloor\!\lfloor} (y)_n^e)_n^e.$

<u>PROOF</u>. Straightforward. $\square$

Let $(E \cup I)_n^e$ be the set of all $(x)_n^e$ for $x \in (E \cup I)_{\omega}$, for some fixed $n \geq 1$. The operations $+_n^e, \cdot_n^e, \underset{n}{\lfloor\!\lfloor}^e$ and $\|_n^e$ are defined on $(E \cup I)_n^e$ as follows:

$$(x +_n^e y) = (x + y)_n^e$$

etc. Thus we obtain a process algebra

$$(E \cup I)_n^e (+, \cdot, \lfloor\!\lfloor, \|)$$

for each $n \geq 1$.

3.6. <u>Bisimulation modulo I on bounded processes</u>.

We want to use Theorem 3.5.2 as a basis for defining $\underset{I}{\rightleftharpoons}$ on processes in the algebra of bounded processes $B^{\infty}(E,I)$. In order to do so, we need the following definition.

3.6.1. <u>DEFINITION</u>. For $p = (p_1, p_2, \ldots) \in B^{\infty}(E,I)$ and $n \geq 1$:

$$(p)_n^e = \lim_{i \to \omega} (p_i)_n^e.$$

The existence of this limit is the very implication of the fact that $p \in B^{\infty}(E,I)$. Indeed, this existence could be used for an alternative definition of $B^{\infty}(E,I)$.

3.6.2. DEFINITION. For $p,q \in B^{\infty}(E,I)$ we define $p \Leftrightarrow_I q$ if

$$\forall n \geqslant 1: \quad (p)_n^e \Leftrightarrow_I (q)_n^e.$$

In this way $\Leftrightarrow_I$ becomes an equivalence relation on $B^{\infty}(E,I)$ but not a congruence.

3.7. $\underline{GB^{\infty}(E,I):\ guarded\ and\ bounded\ processes\ over\ E \cup I.}$

We combine previous observations as follows: let

$$GB^{\infty}(E,I) = G^{\infty}(E,I) \cap B^{\infty}(E,I)$$

where $G^{\infty}(E,I)$ contains all externally guarded processes of $(E \cup I)^{\infty}$.

3.7.1. THEOREM. $\Leftrightarrow_I$ *is a congruence on* $GB^{\infty}(E,I)$.

PROOF. Straightforward. $\square$

We conclude that a factor algebra $GB^{\infty}(E,I)/\Leftrightarrow_I$ exists in which E^{∞} can be isomorphically embedded.

3.7.2. THEOREM. *In* $GB^{\infty}(E,I)/\Leftrightarrow_I$ *each equation of the form*

$$x = t_e(x)$$

with t_e *an* $E(+,\cdot,\lfloor\lfloor,\|)$*-term in which each occurrence of* x *is guarded, possesses a unique solution.*

PROOF. Because $x = t_e(x)$ has already a solution in E^{∞}, the existence of the solution in $GB^{\infty}(E,I)/\Leftrightarrow_I$ is clearly guaranteed.

Now suppose that both p and q solve $x = t_e(x)$. Then $p \Leftrightarrow_I t_e(p)$ and $q \Leftrightarrow_I t_e(q)$, thus arguing with induction on n:

$$(p)_n^e \Leftrightarrow_I (t_e(p))_n^e \Leftrightarrow_I (t_e((p)_{n-1}^e))_n^e \Leftrightarrow_I$$

$$(t_e((q)_{n-1}^e))_n^e \Leftrightarrow_I (t_e(q))_n^e \Leftrightarrow_I (q)_n^e$$

whence $p \Leftrightarrow_I q$. $\square$

4. INTERNAL ACTIONS AS A RESULT OF INTERNAL COMMUNICATIONS

4.1. Algebra of communicating processes (ACP).

We start with describing the structure of the set of atomic actions common
to our examples.

Let $A = E \cup \{\delta\} \cup I \cup H$ be a disjoint partitioning of A, with:

E: *external actions*

I: *internal actions*

H: *hidden or subatomic actions*

δ: *new constant for deadlock (or failure)*.

In our examples we will illustrate how the abstraction theory for $GB^{\infty}(E,I)$
can be applied. To this end we introduce subatomic actions H. These actions
have to communicate (coöperate) in any process execution. A typical example
of subatomic actions is presented by the CSP primitives C!x and C?y. These
actions can communicate, thus yielding an assignment:

$$C!x \mid C?y = (y:=x).$$

The algebra of communicating processes ACP, see [6,7], results from PA by adding
a communication operator "|". On atoms .|. yields atoms, thus we assume that
$a \mid b \in A$ for each $a,b \in A$.

If $a \mid b = \delta$ we say that a and b do not communicate.

In our examples we will have the following properties for .|. on A:

(i) $a \mid \delta = \delta$, $a \mid b = b \mid a$,

(ii) $a \mid b = \delta$ if a or $b \in E \cup I$ *(only subatomic actions communicate)*

(iii) $a \mid b \neq \delta \implies a \mid b \in I$ *(communications lead to internal actions)*

(iv) $a \mid (b \mid c) = (a \mid b) \mid c = \delta$ *(no ternary communications)*
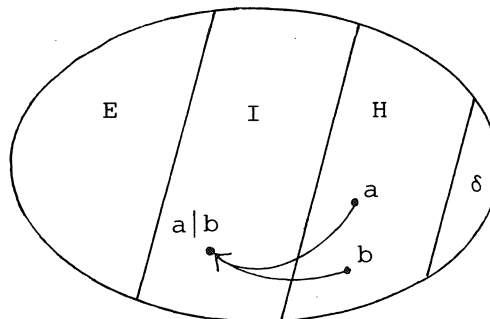
Based on A and $.|.: A \times A \to A$ the following axioms of ACP describe communicating processes. The most important equation is CM1:

$$x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x | y.$$

Here a new term $x|y$ occurs (as compared with the corresponding axiom of PA) which indicates that x and y must share their first step.

Restricting to $(E \cup I)^\infty$ this component $x|y$ is always δ and then disappears, thus leading to equations corresponding to PA.

ACP:

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y).z = x.z + y.z$	A4
$(x.y).z = x.(y.z)$	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
$a \lfloor\!\lfloor x = a.x$	CM2
$(ax) \lfloor\!\lfloor y = a(x \| y)$	CM3
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	CM4
$(ax)|b = (a|b).x$	CM5
$a|(bx) = (a|b).x$	CM6
$(ax)|(by) = (a|b).(x \| y)$	CM7
$(x + y)|z = x|z + y|z$	CM8
$x|(y + z) = x|y + x|z$	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
$\partial_H(x.y) = \partial_H(x).\partial_H(y)$	D4

The operator ∂_H, present for each $H \subseteq A$, has the task of removing unsuccessful communications. In terms of CSP: $C!x | C?y = (y:=x)$, and H will contain both $C!x$ and $C?y$ in order to ensure that in $\partial_H[P_1 \| \ldots \| P_n]$ no (subatomic) action like $C!x$ or $C?y$ is performed unless in appropriate communication with its counterpart.

Just like for PA one can find models $A_\omega(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel,\mid,\delta)$ and $A_n(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel,\mid,\delta)$ for $n \geqslant 1$, and a projective limit $A^\infty(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel,\mid,\delta)$. The substructures containing actions from $E \cup I$ only and without '$\mid$' are just

$$(E \cup I)_\omega(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel),$$

$$(E \cup I)_n(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel) \text{ and}$$

$$(E \cup I)^\infty(+,\cdot,\parallel\!\!\!\!\!\perp,\parallel)$$

as they have been introduced in Section 1. In particular $GB^\infty(E,I)$ can be seen as a substructure of A^∞ $(= (E \cup I \cup H \cup \{\delta\})$).

The format of our applications is as follows: we are given some processes $P_1,\ldots,P_k$ over A and some process Q over E. Q acts as a specification and has to be implemented using $P_1,\ldots,P_k$. In order to do so we may provide some extra processes $R_1,\ldots,R_\ell$ over A (subject to certain constraints) and take

$$Q^* = \partial_H(R_1 \parallel \ldots \parallel R_\ell \parallel P_1 \parallel \ldots \parallel P_k).$$

Now Q^* will be in $GB^\infty(E,I)$ and one has to show that modulo I-bisimulation congruence

$$Q^* \equiv Q.$$

This verifies the correctness of the implementation of Q on the basis of the P_i ($i = 1,\ldots,k$).

4.2. Locations.

In order to understand the conditions that may be imposed on the auxiliary programs $R_1,\ldots,R_\ell$ it is easiest to think of Q as a distributed system and to assume that each atomic act takes place at one of a finite set, say α,β,γ, of *locations*. (Logically a location can be thought of as a finite set of ports.) This leads to a further partitioning of A:

22
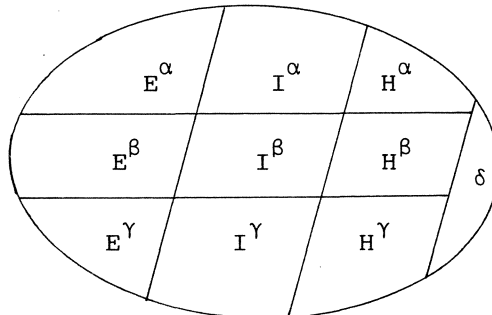
Now communications between subatomic actions of different locations are required to yield δ. Moreover, the auxiliary programs $R_1,\ldots,R_\ell$ may use actions of one location only, i.e. these are local programs that manufacture the distributed system Q^* using the $P_1,\ldots,P_k$ which connect the different locations.

In the terminology of communication protocols: the $P_1,\ldots,P_k$ are media, the $R_1,\ldots,R_\ell$ are senders, receivers etc., and Q is a protocol to be implemented. Proving

$$Q \equiv Q^* = \partial_H (R_1 \| \ldots \| R_\ell \| P_1 \| \ldots \| P_k)$$

then stands for showing the correctness of the implementation of the protocol.

5. EXAMPLE: CONNECTING TWO BAGS

For two locations α and β and a (finite) set of data D, the bag $B^{\alpha\beta}$ is a channel which receives data d at α (action d^α for $d \in D$) and delivers these data at β (action d^β). $B^{\alpha\beta}$ will deliver all data it has been offered but in arbitrary order.

The action alphabet involved is just $D^\alpha \cup D^\beta$. The bag is described (according to [7]) by the following fixed point equation:

$$B^{\alpha\beta} = \sum_{d \in D} d^\alpha (d^\beta \| B^{\alpha\beta}).$$
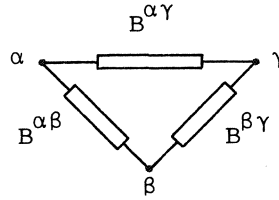
Now consider a network consisting of two bags:
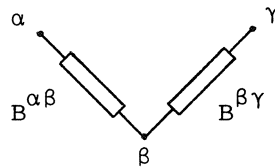
The bags are connected because at location β the communication function is defined as follows:

$$d^\beta | d^\beta = \underline{d}^\beta.$$

This leads to the following alphabet of actions:

location	external actions	internal actions	subatomic actions
α	D^α	$\emptyset$	$\emptyset$
β	$\emptyset$	$\underline{D}^\beta$	D^β
γ	D^γ	$\emptyset$	$\emptyset$
	$E = D^\alpha \cup D^\gamma$	$I = \underline{D}^\beta$	$H = D^\beta$

The construct $\partial_H(B^{\alpha\beta} || B^{\beta\gamma})$ will now realise a bag-connection between α and γ.

In order to see that $\partial_H(B^{\alpha\beta} || B^{\beta\gamma})$ acts like a bag again, note that a communication $\underline{d}^\beta = d^\beta | d^\beta$ at β describes passing a value d from $B^{\alpha\beta}$ to $B^{\beta\gamma}$.

To verify formally that $\partial_H(B^{\alpha\beta} || B^{\beta\gamma})$ is externally equivalent with $B^{\alpha\gamma}$ one first derives within ACP a fixed point equation for $B = \partial_H(B^{\alpha\beta} || B^{\beta\gamma})$:

$$B = \sum_{d \in D} d^\alpha ((\underline{d}^\beta . d^\gamma) || B).$$

The proof rests on induction on n, working modulo n. We omit the straightforward but lengthy details.

Because B satisfies a guarded fixed point equation, B is in $GB^\infty(E,I)$. We will now restrict our attention to $GB^\infty(E,I)$ where both $B^{\alpha\gamma}$ and B exist. Working modulo $\underset{I}{\Longleftrightarrow}$ we observe that

$$B = \sum_{d \in D} d^\alpha (\underline{d}^\beta . d^\gamma || B) = \sum_{d \in D} (d^\alpha \underline{d}^\beta d^\gamma \lfloor\!\lfloor B) \underset{I}{\Longleftrightarrow} \quad (*)$$

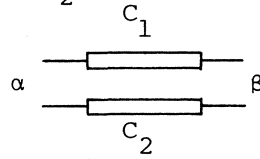$$\sum_{d \in D} (d^\alpha d^\gamma \lfloor\!\lfloor B) = \sum_{d \in D} d^\alpha (d^\gamma || B).$$

(*): this equivalence follows from $d^\alpha \underline{d}^\beta d^\gamma \underset{I}{\Longleftrightarrow} d^\alpha d^\gamma$ and the fact that $\underset{I}{\Longleftrightarrow}$ is a congruence on $GB^\infty(E,I)(+,\cdot,\lfloor\!\lfloor,||)$.

Now from Theorem 3.7.2 it follows that $B \underleftrightarrow{}_I B^{\alpha\gamma}$ as they are both solutions of the guarded E-equation

$$x = \sum_{d \in D} d^{\alpha}(d^{\gamma} \| x).$$

6. EXAMPLE: A SIMPLE COMMUNICATION PROTOCOL

Here we are dealing with two locations α and β. These locations are connected by two media C_1 and C_2:

The problem is to implement a communication protocol T which transmits data $d \in \{0,1\}$ from α to β and acknowledges at α when data have been received at β.

The external alphabet E is $\{0,1,\underline{0},\underline{1},a\}$ where $0,1,a$ are actions at location α and $\underline{0},\underline{1}$ at location β. Then T should externally have the behaviour specified by this recursion equation:

$$T = (0.\underline{0} + 1.\underline{1})a.T.$$

In a figure:
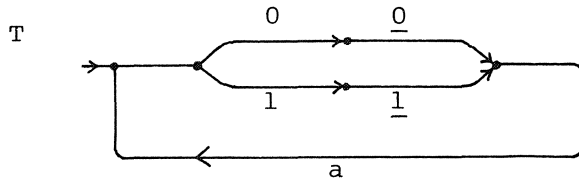
The channels C_1 and C_2 are as follows:

$$C_1 = (s.\underline{s} + t.\underline{t})C_1$$

$$C_2 = \underline{u}.u.C_2.$$

Here s,t,u are subatomic actions at location α and $\underline{s},\underline{t},\underline{u}$ are subatomic actions at location β. Thus C_1 permits $\alpha \to \beta$ transmission of s or t and C_2 permits $\beta \to \alpha$ transmission of a single signal u.

We will implement T by means of two auxiliary processes: A, a sender using actions for location α only and B, a receiver using only actions for location β. The actions of A and B are the following:

A

0	*receive* 0
1	*receive* 1
a	*acknowledge* a
s	*offer* s *to* C_1
t	*offer* t *to* C_1
u	*expect* u *from* C_2

B

$\underline{0}$	*send* 0 *outside*
$\underline{1}$	*send* 1 *outside*
$\underline{s}$	*expect* s *from* C_1
$\underline{t}$	*expect* t *from* C_1
$\underline{u}$	*offer* u *to* C_2

Internal actions are:

s°	A *signals* s *to* C_1
t°	A *signals* t *to* C_1
u°	A *receives* v *from* C_2

$\underline{s}$°	B *receives* s *from* C_1
$\underline{t}$°	B *receives* t *from* C_1
$\underline{u}$°	B *signals* v *to* C_2

The architecture of the implementation of T is as follows:

$$\partial_H (A \| B \| C_1 \| C_2)$$

where $H = \{s,t,u,\underline{s},\underline{t},\underline{u}\}$. Moreover $E = \{0,1,\underline{0},\underline{1},a\}$ and $I = \{s°,t°,u°,\underline{s}°,\underline{t}°,\underline{u}°\}$. The aim is to have A and B such that working in $GB^\infty(E,I)/\underset{I}{\leftrightarrow}$:

$$T = \partial_H (A \| B \| C_1 \| C_2).$$

We choose A and B as follows:

$$A = (0.s + 1.t)uaA$$

$$B = (\underline{s}.\underline{0} + \underline{t}.\underline{1})\underline{u}B.$$

Now working in $(E \cup I \cup H)^\infty$ and writing T* for $\partial_H (A \| B \| C_1 \| C_2)$ one easily proves:

$$T^* = (0.s.\underline{s}.\underline{0} + 1.t.\underline{t}.\underline{1})\underline{u}uT^*.$$

Working modulo $\underset{I}{\leftrightarrow}$ in $GB^\infty(E,I)$ we see that:

$$T^* = (0.s.\underline{s}.\underline{0}.\underline{u}.u + 1.t.\underline{t}.\underline{1}.\underline{u}.u)aT^* \underset{I}{\leftrightarrow} (0.\underline{0} + 1.\underline{1})aT^*$$

because $0ss\underline{0}\underline{u}u \underset{I}{\leftrightarrow} 0\underline{0}$ and $1tt\underline{1}\underline{u}u \underset{I}{\leftrightarrow} 1\underline{1}$.

Now T* and T satisfy the same guarded E-fixed point equation and thus coincide in $GB^\infty(E,I)/\underset{I}{\leftrightarrow}$ by virtue of Theorem 3.7.2.

7. DISCUSSION AND CONCLUDING REMARKS

7.1. Process algebra as used in this paper has three sources. MILNER [13]
works with an algebraic formalism in which the operators of process algebra
are present (except $\parallel$). In DE BAKKER & ZUCKER [3,4] topological methods are
used to provide a semantics for systems from MILNER [13]. In [5] the present
authors provide an algebraic reconstruction of this semantics as a projecti-
ve limit (A^{∞} of the present paper). In [6] ACP was introduced, and in [7]
value passing in ACP has been treated.

Concerning the abstraction mechanism that we use: bisimulation occurs
earlier in MILNER [14] and PARK [15] (however not in the presence of internal
actions). The use of bisimulation as an abstraction mechanism might be a
novelty of the present paper.

7.2. On various concepts of processes.

The notion of a process is inherently vague up to now. Our algebraic approach
is best understood as an axiomatic one in which the nature of processes is
left vague but their properties can be discussed. The 'standard' model A^{∞}
is in our view nothing more than some model of ACP. It must be admitted how-
ever that A^{∞} combines several particular features which give it a special
status at the moment.

Milner's approach in [13] is rather modeltheoretic in spirit, processes
as terms or congruence classes come first and equations are derived afterwards.
These equations then are more appealing to the human eye than the semantics
thus giving CCS an axiomatic flavour as well.

Quite different points of view occur in PRATT[16] and BROCK & ACKERMAN [8]．
There processes occur as scenario's. Scenario's provide a nontrivial generali-
zation of trace theory (to be discussed below as well). Scenario's allow true
simultaneity of actions, even at an atomic level. The price to be paid for
this is that scenario's are a hard medium for explicit calculation.

It seems to be the case that ACP and the likes are about "sequential"
concurrent process theory and that scenario's are aiming one level higher,
at the description of true concurrency. The key point here is that a substan-
tial range of phenomena about concurrency and communication can already be
described at the "sequential" level (as do CSP, CCS, and also our ACP).

The mathematics of that range requires substantial attention at the present stage of development. For this purpose we restrict our attention to ACP, and leave the development of more sophisticated notions (like the conceptually superior scenario's) for later work.

In ARNOLD [1], BACK & MANNILA [2] and REM [17] one finds trace theories of processes. In a trace theory the semantics of a process is a set of (possible execution traces. The advantage of trace theory is that it is mathematically simple. The disadvantage is its less sophisticated treatment of communication refected in an absence of a proper treatment of deadlock and termination, as BACK & MANNILA [2] explicitly state.

7.3. Abstraction theory.

This issue is of considerable importance. (In PRATT [16] it is argued that the presence of workable abstraction tools is a fundamental criterion for any algorithmic theory of processes.) Especially in specification and standardization of communication protocols, abstraction from low level implementation details is essential. The mathematics of these matters is at present not well-understood.

An important contribution is observational equivalence as defined by MILNER [13] in his pioneering study on CCS. Our present paper aims at an algebraic background for Milner's work.

Many problems are left. We mention two of these:
(1) How to eliminate internal steps from this process:

$$a(ib + ic).$$

HENNESSY [9] and MILNE [12] discuss these matters using an operator $\oplus$.
In our view a proper algebraic understanding (of $\oplus$) is not yet present however.
(2) Secondly, processes may well contain infinitely long traces of internal steps. Which hiding mechanisms apply here? Stated differently: who is x, with $x = a + ix$?

Finally it should be said that it is quite mysterious to us how notions of fairness and real-time computation can be added to the present framework. Especially real-time computation may require an entire redesign of the formalism.

8. APPENDIX: AN ALGEBRAIC CHARACTERIZATION OF BISIMULATION MODULO INTERNAL
   STEPS FOR FINITE PROCESSES

8.0. Introduction.

In the first part of this note we will give for finite and acyclic process graphs $(\mathcal{G}_A$, see Section 2.1) a characterization of the notion of bisimulation modulo internal steps as introduced in section 3.1.

This characterization will take the following form: two process graphs $g_1, g_2 \in \mathcal{G}_A$ are in bisimulation modulo I ($g_1 \underset{I}{\leftrightarrow} g_2$) iff g_1, g_2 can be reduced to identical 'normal' graphs, via a certain simplification procedure defined on graphs. The normal graph of a graph g is not only bisimulation equivalent mod. I with g, but is also the unique minimal (w.r.t. the number of nodes) graph with that property.
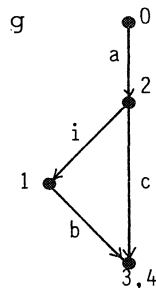
8.0.1. Example.
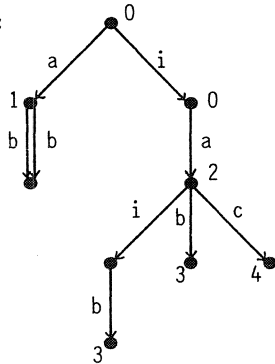
Let g_1 be: and g_2:
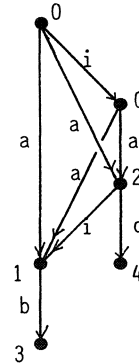
Then $g_1 \underset{I}{\leftrightarrow} g_2$ as indicated by the numbering of the nodes (node i in g_1 is in the intended bisimulation relation connected to all nodes i in g_2 and vice versa).

Both g_1 and g_2 reduce to the normal graph g:

In the second part of this note we derive as a corollary of the characterization of bisimulation for process graphs, a complete axiomatization for process terms with internal steps (more precisely, for the congruence $\underset{I}{\leftrightarrow}$ in $G_\omega(E,I)$, see 3.4). This axiomatization is in addition to A1-5,M1-4 (the axioms of PA) given by the axioms

$i = j$	I1
$xi = x$	I2
$ix + x = ix$	I3
$a(ix + y) + ax = a(ix + y)$	I4

(Here $a \in E$, the set of external actions, and $i,j \in I$, the set of internal actions.)

Apart from minor differences, these axioms I1-4 occur as the 'τ-laws' in MILNER [13] (Theorem 7.13) and in HENNESSY & MILNER [10]. This is not surprising, since it will not be hard to prove that the notion of 'observational equivalence' (in the presence of τ's) coincides with the notion of bisimulation (modulo I), at least for finite processes. Furthermore, our concept 'externally guarded' is called in [13], 'stable'.

The treatment below via process graphs makes the axioms I1-4 very perspicuous. The axiom I3 corresponds to the fact that in a process graph g the subgraph of the form g_1 may be replaced by g_2, while preserving bisimulation mod. I:

g_1 with arrows u, i, u g_2 with arrows i, u $(i \in I, \ u \in E \cup I)$
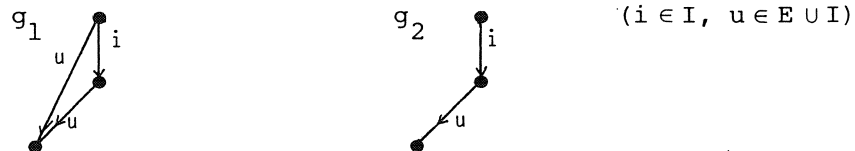
Likewise I4 corresponds to the replaceability of

g_1 with arrows u, u, i by g_2 with arrows u, i .
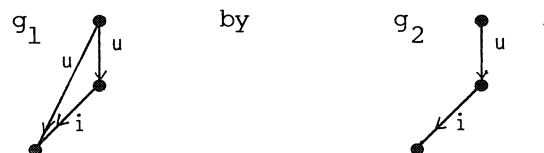
8.1. A characterization of bisimulation modulo I on $\mathcal{G}_A$.

In the sequel, up to Theorem 8.2.2, we suppose for simplicity that $I = \{i\}$. Before defining the reduction procedure for process graphs, we need some terminology.

8.1.1. DEFINITION. Let $g \in \mathcal{G}_A$.

(i) A *subgraph* g' of g consists of an arbitrary subset of the set of edges of g (plus their labels $\in$ A) together with the nodes belonging to these edges.

(ii) Let $s \in \text{NODES}(g)$. Then g_s is the subgraph of g consisting of *all* nodes and edges of g which are 'below' s (including s itself) in the obvious sense. We will call g_s a *full* subgraph.
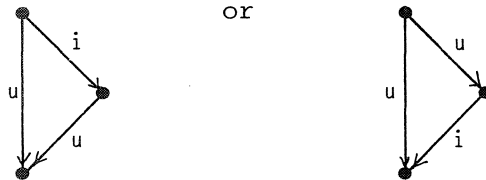
(iii) An *arc* in g is a subgraph of the form
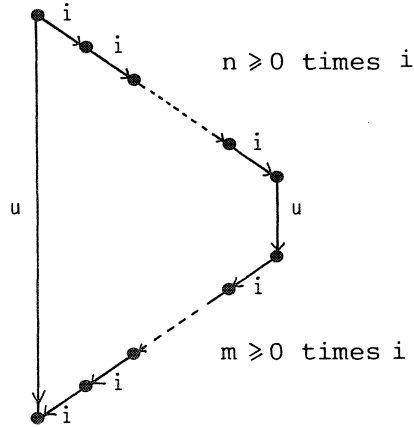
where $u \in E \cup I$. The u-edge at the left is called the *primary* edge of the arc, the other edges are called secondary edges.

(iv) If in (iii) $n = m = 0$ the arc has the form and is called a *double edge*.

(v) If in (iii) $n + m = 1$ the arc has the form

and is called an *elementary arc*.

We will now define a simplification procedure on process graphs in $\mathcal{G}_A$, consisting of three basic simplification rules (or 'reduction' rules).

One of these rules is stated in terms of *identifying a pair of nodes* s_1, s_2, which intuitively means that s_1, s_2 are glued together; e.g. g_1 yields g_2 after identifying s_1, s_2 in the following example:
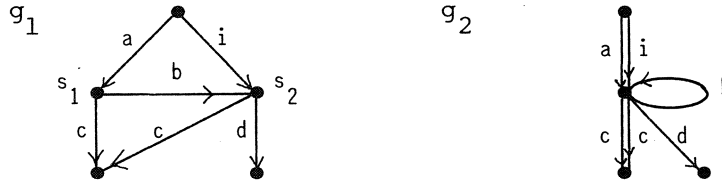
(Note that $g_2 \notin \mathcal{G}_A$.)

8.1.2. DEFINITION. On the set $\mathcal{G}_A$ of process graphs we define the following three reduction relations:

[i] *Sharing.* Let $g \in \mathcal{G}_A$ contain nodes s_1, s_2 such that g_{s_1} is isomorphic to g_{s_2}. Then g reduces to g' where s_1, s_2 are identified.
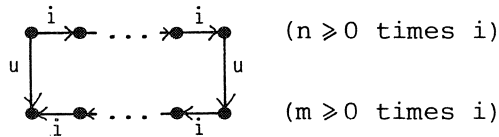
(Note that again $g' \in \mathcal{G}_A$, by the isomorphism condition.)

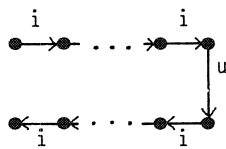[ii] *Removal of deterministic internal steps.*

If $s_1 \bullet \xrightarrow{i} \bullet s_2$ occurs in g and the outdegree of s_1 is *one* (i.e. the displayed i-step is a 'deterministic' internal step), then s_1, s_2 may be identified after removal of the i-edge.

[iii] *Arc reduction.* In an arc the primary edge may be deleted. I.e. the subgraph
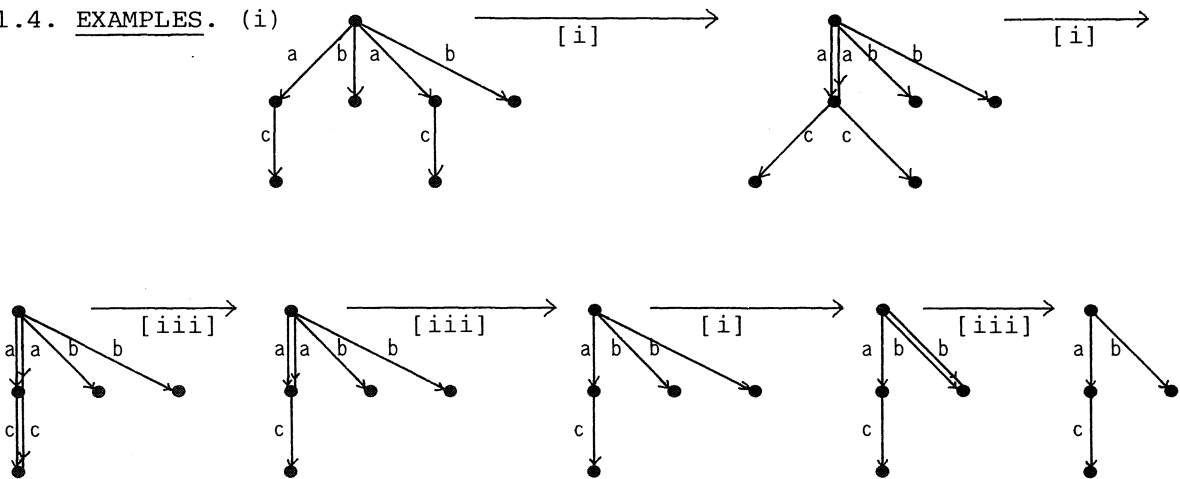

may be replaced by

8.1.3. <u>NOTATION</u>. If g_1 reduces to g_2 by one application of [i] sharing, [ii] deterministic i-removal, or [iii] arc reduction we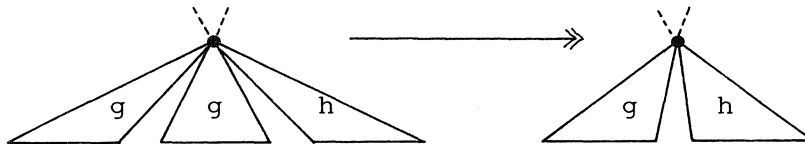 write $g_1 \longrightarrow g_2$. (According to which of the three, we write sometimes $g_1 \xrightarrow[{[i]}]{} g_2$, etc.) If g_1 reduces to g_2 by an arbitrary number of these reduction 'steps', possibly zero, we write $g_1 \longrightarrow\!\!\!\!\gg g_2$. (So $\longrightarrow\!\!\!\!\gg$ is the transitive reflexive closure of $\longrightarrow$.)

8.1.4. <u>EXAMPLES</u>. (i)

(ii) Likewise one proves that

(iii)

(iv)

The next theorem states the *soundness* of process graph reduction w.r.t bisimulation modulo I. The proof is routine.

8.1.5. <u>THEOREM</u>. *Let* $g_1, g_2 \in \mathcal{G}_A$. *Then* $g_1 \longrightarrow\!\!\!\!\! \gg g_2$ *implies* $g_1 \underset{I}{\Longleftrightarrow} g_2$. $\square$

In order to prove the *completeness* of process graph reduction we first prove the following fact.

8.1.6. <u>THEOREM</u>. *(i) Every process graph reduction* $g_1 \to g_2 \to \ldots$ *must terminate eventually.*

(ii) Process graph reductions are confluent (or: have the Church-Rosser property). I.e. if $g_1 \longrightarrow\!\!\!\!\! \gg g_2$ *and* $g_1 \longrightarrow\!\!\!\!\! \gg g_3$, *then for some* g_4 *we have* $g_2 \longrightarrow\!\!\!\!\! \gg g_4$ *and* $g_3 \longrightarrow\!\!\!\!\! \gg g_4$.

$$
\begin{array}{ccc}
 & g_1 & \\
\swarrow & & \searrow \\
g_2 & & g_3 \\
\searrow & & \swarrow \\
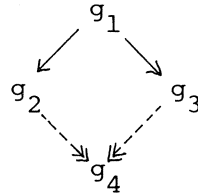 & g_4 &
\end{array}
$$

(iii) Every $g \in \mathcal{G}_A$ *can be reduced to a* <u>normal</u> *graph* g' *(i.e. a process graph from which no further reduction steps are possible), which is* <u>unique</u> *in that respect.*

<u>PROOF</u>. (i) is immediate since in every reduction step the number of nodes or the number of edges (or both) decreases.

(iii) By (i) every g reduces to a normal graph. By (ii) the unicity follows (two normal graphs can only have a common reduct if they coincide).

The hard part of this proof is to establish (ii). By the well-known Lemma of Newman (see e.g. [11]) it suffices, in view of the termination property proved in (i), to establish the weaker form of (ii) where "$g_1 \twoheadrightarrow g_2$ and $g_1 \twoheadrightarrow g_3$" is replaced by "$g_1 \rightarrow g_2$ and $g_1 \rightarrow g_3$":

$$
\begin{array}{ccc}
 & g_1 & \\
\swarrow & & \searrow \\
g_2 & & g_3 \\
\searrow & & \swarrow \\
 & g_4 &
\end{array}
$$

(The 'Weak Church-Rosser' property.)

According to whether $g_1 \rightarrow g_2$ and $g_1 \rightarrow g_3$ are [i]-,[ii]- or [iii]-reduction steps (see Definition 8.1.2), we must check six cases:

(1) [i]- versus [i]-reduction

(2) [i]- versus [ii]-reduction

(3) [i]- versus [iii]-reduction

(4) [ii]- versus [ii]-reduction

(5) [ii]- versus [iii]-reduction

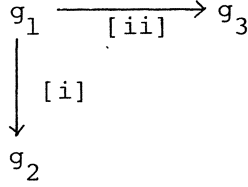(6) [iii]- versus [iii]-reduction.

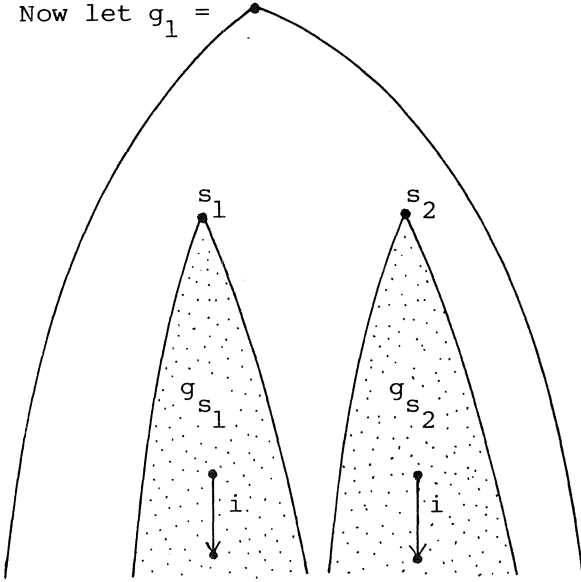The cases not involving [i] are easy. Case (4) is the most easy of them. A typical example of (5) is
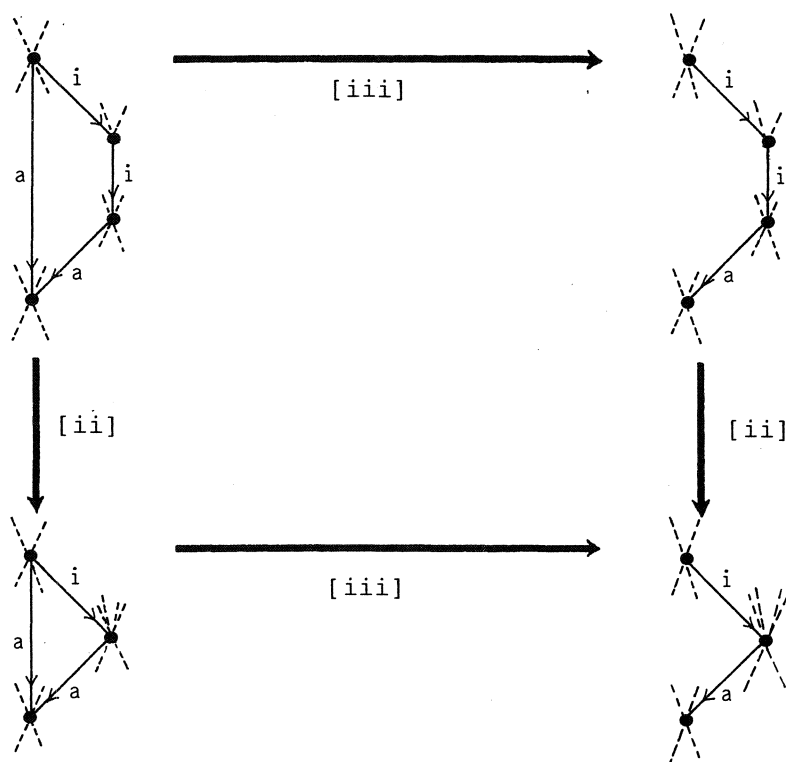
A typical example of (6) is
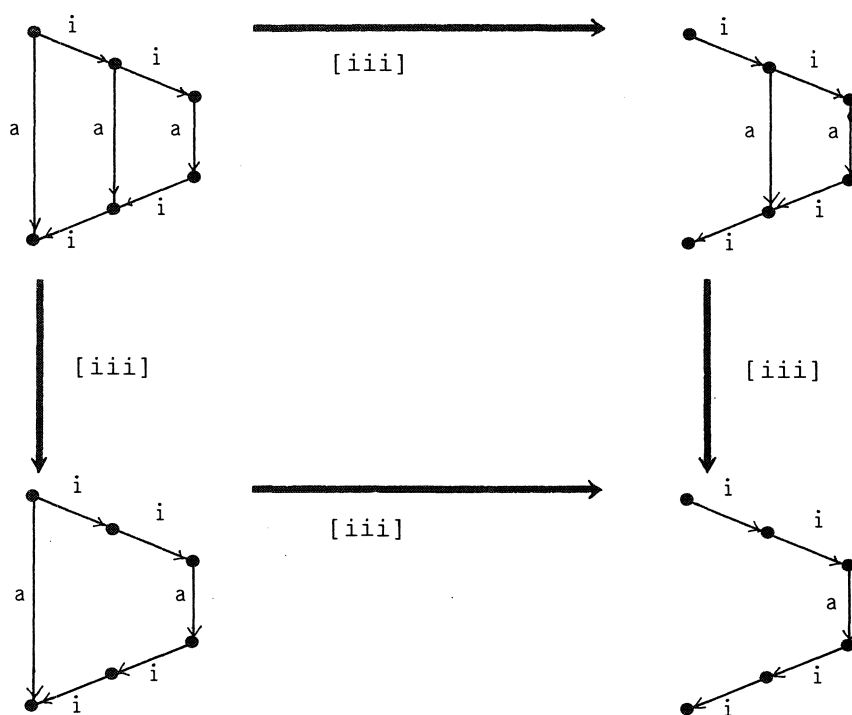
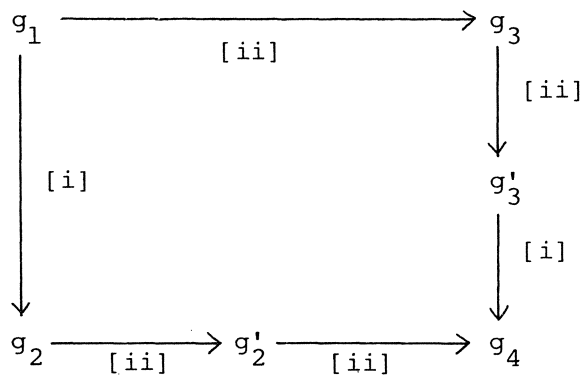The proofs for the cases (1),(2),(3) have a common feature. We will start with case (2):

$$g_1 \xrightarrow{\quad [ii] \quad} g_3$$
$$\downarrow [i]$$
$$g_2$$

Now let $g_1 =$

where g_{s_1}, g_{s_2} are isomorphic and the $g_1 \rightarrow g_2$ step identifies s_1, s_2. Suppose $g_1 \rightarrow g_3$ contracts an i-step in, say, g_{s_1}. For the sake of exposition, let g_{s_1} and g_{s_2} first be <u>disjoint</u> subgraphs of g_1. Then there is no problem in finding a common reduct g_4: first contract in g_3 the 'corresponding' i-step in g_{s_2} (since g_{s_1}, g_{s_2} are isomorphic, this corresponding i-step is there), and now a [i]-reduction can again be applied:

$$
\begin{array}{ccccc}
g_1 & \xrightarrow{\quad [ii] \quad} & & & g_3 \\
\downarrow{[i]} & & & & \downarrow{[ii]} \\
& & & & g_3' \\
& & & & \downarrow{[i]} \\
g_2 & \xrightarrow{[ii]} & g_2' & \xrightarrow{[ii]} & g_4
\end{array}
$$

In the general case in which g_{s_1}, g_{s_2} need not be disjoint, we proceed as follows. Let $\phi: g_{s_1} \longrightarrow g_{s_2}$ be the given isomorphism. (See figure below.)

<u>Example</u>: g_1 is

$$g_{t_1} \qquad g_{t_2} \qquad g_{t_3} \qquad g_{t_4}$$
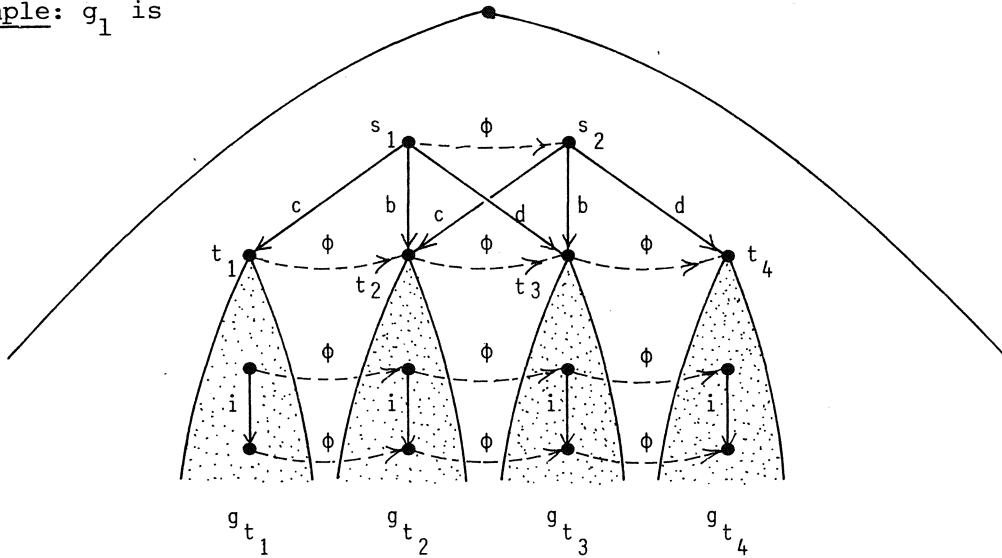
Here the g_{t_i} (i = 1,..,4) are isomorphic; they may overlap.

Now in order to 'compensate' for the reduction $g_1 \xrightarrow[\text{[ii]}]{} g_3$, which

contracts say the i-step in g_{t_2}, <u>all</u> the 'corresponding' i-steps have to be

contracted, where 'corresponding' refers to the steps related by ϕ. That is,

in the example above the displayed i-steps in g_{t_j} (j = 1,3,4) have to be con-

tracted in order to reestablish the isomorphism between g_{s_1}, g_{s_2} and to make

an identification of s_1, s_2 in g_3' again possible:
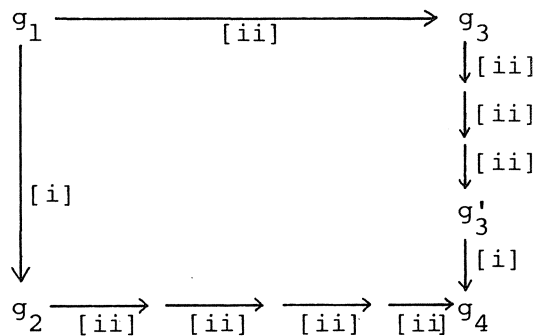

<u>Case (1)</u> can be treated similarly to case (2). This is also true for case (3),

but here we have to make one additional remark: consider a situation as in

the last example, where the i-step is now an arc (see next figure).
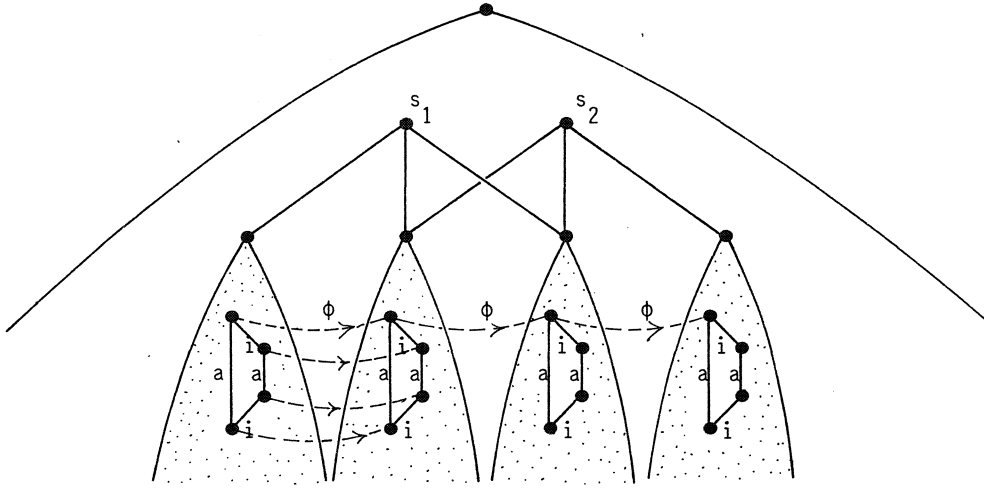
Again we want to leave out the primary steps of all these by ϕ to each other related arcs. This is only possible if ϕ does not confuse primary and secondary edges in an arc; for otherwise removal of the primary step in one arc could destroy another arc in the ϕ-orbit. However, for our acyclic process graphs it is clear that this phenomenon does not occur, in a nontrivial arc (i.e. an arc which is not a 'double edge').

(Remark: indeed this confusion of primary and secondary steps in related arcs may be the case if cycles would be permitted in our process graphs; consider e.g.:
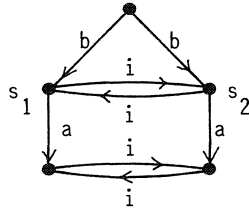
Now removal of the left a-edge destroys the other arc, of which the right a-edge is the primary edge.) □

In order to prove the characterization theorem 8.1.10, we must prove that two normal process graphs which are in bisimulation modulo I, must be identical.

8.1.7. <u>DEFINITION</u>. Let $g \in \mathcal{G}_A$.

(i) A bisimulation mod.I between g and itself, is called an *autobisimulation*.

(ii) g is a *rigid* process graph if it can only be in autobisimulation with itself via the identity relation.

8.1.8. <u>EXAMPLE</u>. The following process graph is not rigid since it admits the displayed nontrivial autobisimulation:
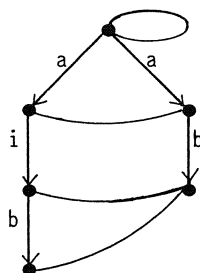
8.1.9. <u>THEOREM</u>. *(i) Normal process graphs are rigid.*

(ii) If g_1, g_2 are normal process graphs and $g_1 \underset{I}{\leftrightarrow} g_2$, then g_1 and g_2 are identical.

<u>PROOF</u>. We prove (i),(ii) simultaneously. In order to do so, we use the abbreviations

R(n): 'a normal process graph with n nodes is rigid' (n ⩾ 1),

U(n): 'two normal process graphs g_1 and g_2, both with less than n nodes, such that $g_1 \underset{I}{\leftrightarrow} g_2$, are identical' (n ⩾ 2).

With induction on n ⩾ 1 we will prove the assertion R(n) & U(n + 1).

<u>Basis</u>. R(1) & U(2): trivial.

<u>Induction step</u>. Suppose, as induction hypothesis, that R(k) & U(k + 1) is proved for 1 ⩽ k ⩽ n. To prove: R(n + 1) & U(n + 2). First we prove R(n + 1).

Consider a normal g with card(g) (= cardinality of NODES(g)) = n + 1, and suppose g is not rigid. Then g contains distinct nodes s,t that bisimulate mod.I (i.e. $g_s \underset{I}{\leftrightarrow} g_t$). Clearly g_s, g_t are as subgraphs of the normal graph g, again normal.

<u>Claim</u>: one of s,t, say s, must be the root of g. For if not then both card(g_s), card(g_t) < n + 1, hence U(n + 1) (part of the induction hypothesis) applies to g_s, g_t, yielding $g_s \equiv g_t$. But then g would be not normal as it contains two nodes with isomorphic full subgraphs.

So let s be the root of g (see figure below) and consider a path π from

s to t. This path can only contain internal steps, otherwise via a simple argument there would be also an external step below t, hence 2 external steps below s, hence 2 external steps below t, hence 3,4,... etc.
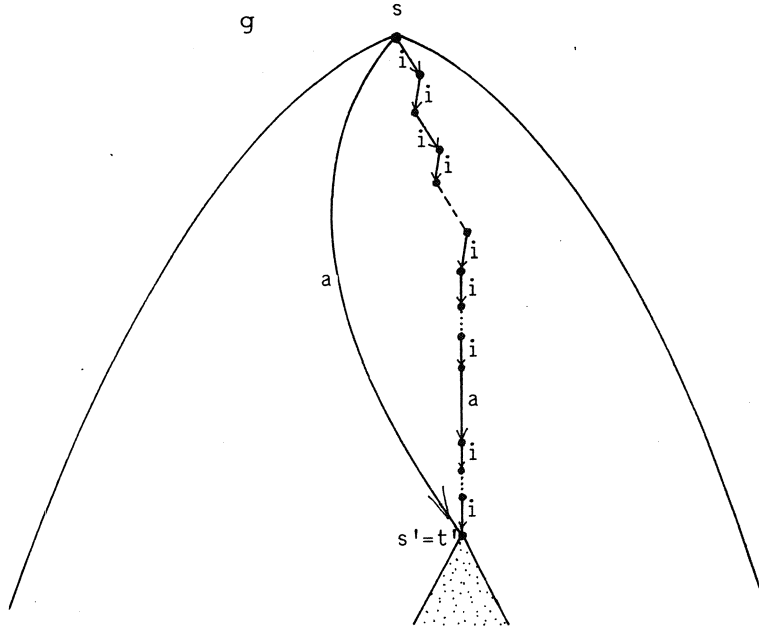
Since g is a normal graph, the first i-step of π cannot be deterministic. So g is
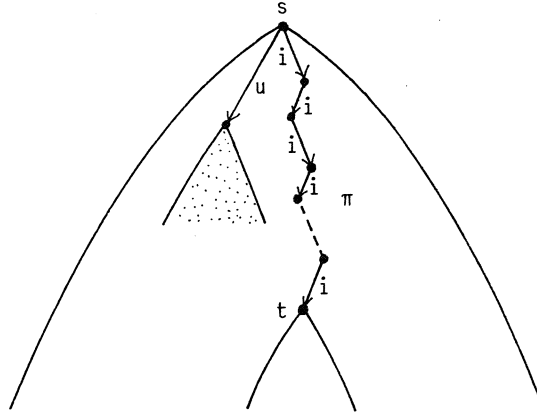
Here $u \in E \cup I$. Say $u = a \in A$. (The case $u = i \in I$) is similar.) Then since $g = g_s \overset{\leftrightarrow}{\rule{0pt}{0pt}}_I\, g_t$, g_t must contain a path as indicated:
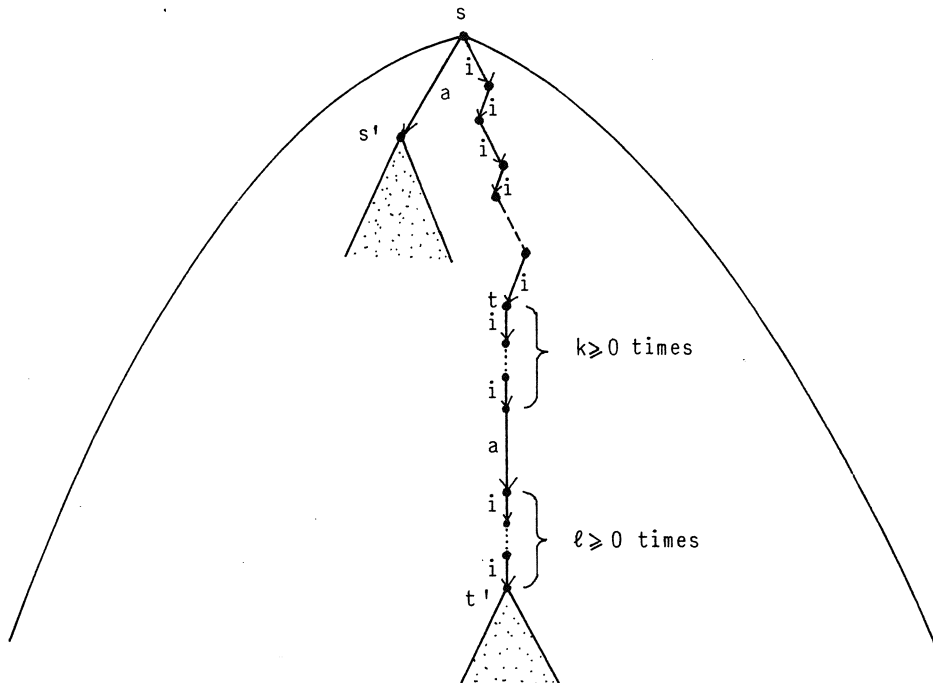
where $g_{s'} \overset{\leftrightarrow}{\rule{0pt}{0pt}}_I\, g_{t'}$, so by another appeal to U_{n+1} we have in fact a coinciden-ce of s',t'. that is, g has the form:

But then g contains an arc, in contradiction with the assumption that g is normal. This proves $R(n+1)$.

Next to prove $U(n+2)$. So let g_1, g_2 be two normal graphs, $g_1 \overset{\longleftrightarrow}{}_I g_2$ and $\text{card}(g_1), \text{card}(g_2) \leqslant n+1$. By $R(n+1)$ therefore, g_1 and g_2 are rigid. This entails that the given bisimulation R between g_1 and g_2 must be in fact a bijection. Therefore we may suppose $\text{card}(g_1) = \text{card}(g_2) = n+1$. The remainder of the argument that $g_1 \equiv g_2$ is an application of similar arguments as before. □

Now we can prove the 'completeness' of process graph reduction, that is:

8.1.10. __THEOREM__. *Let* $g_1, g_2 \in \mathcal{G}_A$. *Then the following are equivalent:*

(i) $g_1 \overset{\longleftrightarrow}{}_I g_2$

(ii) g_1 *and* g_2 *reduce to the same normal process graph.*

(iii) g_1 *and* g_2 *can be converted to each other by means of the process graph reductions in Definition 8.1.1.*

__PROOF__. (ii) $\Leftrightarrow$ (iii) follows from Theorem 8.1.6. Further, (ii) $\Rightarrow$ (i) follows from Theorem 8.1.5. It remains to prove (i) $\Rightarrow$ (ii), or: not(ii) $\Rightarrow$ not(i). So suppose g_1 and g_2 reduce to unequal normal graphs g_1', g_2' respectively. By Theorem 8.1.9, $g_1' \overset{\longleftrightarrow}{}_I g_2'$; hence by Theorem 8.1.5, $g_1 \overset{\longleftrightarrow}{}_I g_2$. □

8.2. An axiomatization for the congruence $\underleftrightarrow{}_I$ on $G_\omega (E,I)$.

In order to transpose this characterization of bisimulation for process graphs to process terms, we must first note that while some graphs obviously correspond (in a direct way) to terms, e.g.

          corresponds to (ac + bd)e,
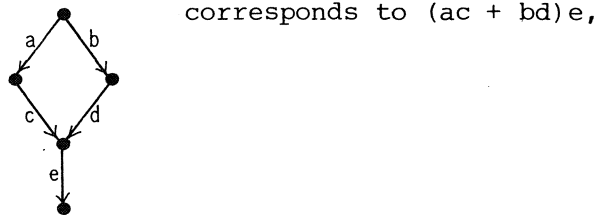
not every graph corresponds to a term; the simplest such graph is
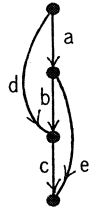
For use in Section 8.2 we will now analyse arc reductions in terms of *elementary arc reductions* as defined in 8.1.1.
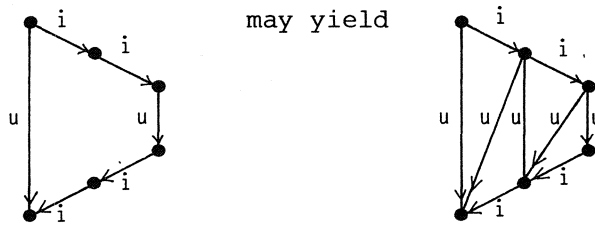
8.1.11. PROPOSITION. *Let* $g_1, g_2 \in \mathcal{G}_A$. *Then* $g_1 \underleftrightarrow{}_I g_2$ *iff* g_1 *and* g_2 *are interconvertible by means of the following reductions:*

[i]   *sharing (as in Definition 8.1.2)*

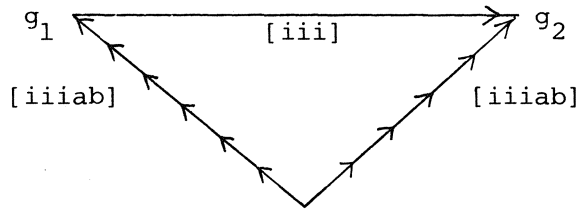[ii]   *removal of deterministic i-steps (as in Definition 8.1.2)*

The elementary arc reductions:

[iiia]

[iiib]

[iiic]
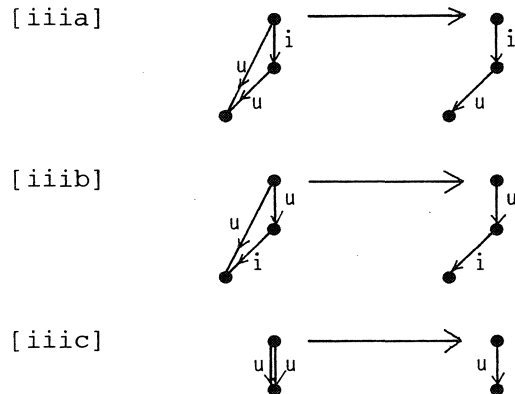
PROOF. Every arc can be filled up with elementary arcs, e.g.:

may yield

Therefore every arc reduction $g_1 \xrightarrow{[iii]} g_2$ can be replaced by a conversion consisting of elementary arc reductions of type [iiiab] (= [iiia] or [iiib]):

We recall the definition in 2.1 of $[\![\ ]\!]$ which assigns to a graph g a term $[\![g]\!]$, essentially by making g into a tree and then taking the term corresponding to this tree. Thus, if

$$g = \quad , \text{ then } [\![g]\!] = dc + a(bc + e).$$
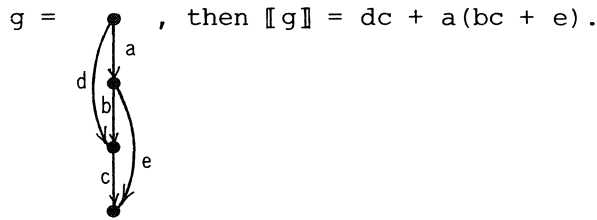
8.2.1. THEOREM. *Bisimulation (modulo I) equivalence on* A_ω *is completely axiomatized by the following equations:*

$$ix \underset{I}{\Longleftrightarrow} x \qquad\qquad I*$$

$$c[i] \underset{I}{\Longleftrightarrow} c[j] \qquad\qquad I1$$

$$c[xi] \underset{I}{\Longleftrightarrow} c[x] \qquad\qquad I2$$

$$c[ix + x] \underset{I}{\Longleftrightarrow} c[ix] \qquad\qquad I3$$

$$c[ax + a(ix + y)] \underset{I}{\Longleftrightarrow} c[a(ix + y)] \qquad\qquad I4$$

Here c[ ] *is an arbitrary* +,·-*context.*

PROOF. Soundness of the axioms is clear. For the completeness, let $t_1, t_2 \in A_\omega$ and $t_1 \leftrightarrow_I t_2$. Consider $g_1, g_2 \in \mathcal{G}_A$ such that $[\![g_1]\!] = t_1$ and $[\![g_2]\!] = t_2$. Then $g_1 \leftrightarrow_I g_2$. By Proposition 8.1.11, g_1 and g_2 are interconvertible via reduction steps [i], [ii], [iiia,b,c]. These reduction steps are transfered by by $[\![\ ]\!]$ into applications of the axioms in question.
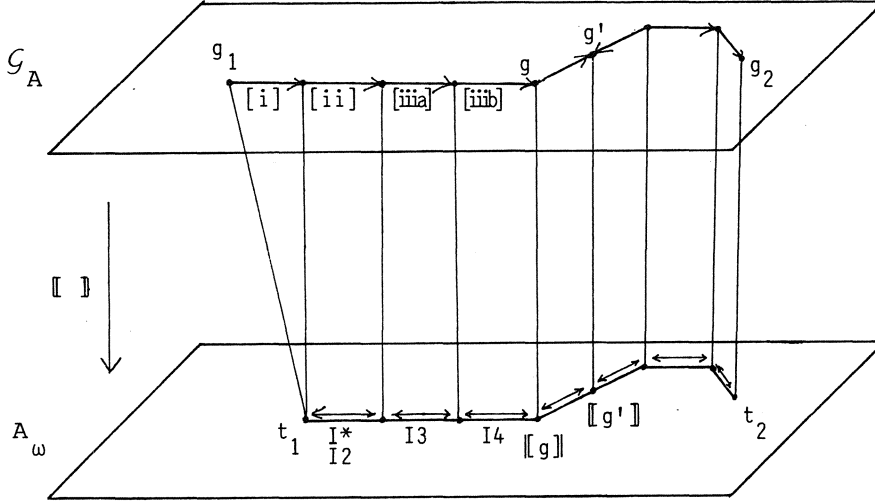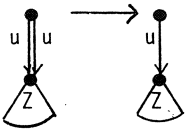

Namely, let $g \longrightarrow g'$ by:

[i]   sharing. Then $[\![g]\!] = [\![g']\!]$.
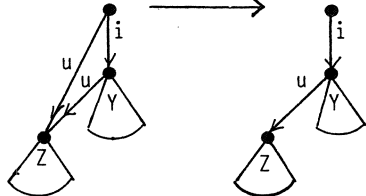
[ii]   contraction of a deterministic i-step. Then either

$$[\![g]\!] = ix \text{ and } [\![g']\!] = x,$$

or there is some $+,\cdot$-context $c[\ ]$ such that

$$[\![g]\!] = c[xi] \text{ and } [\![g']\!] = c[x].$$

[iiic]   . Then $[\![g]\!] = c[uz + uz] = c[x + x]$ and $[\![g']\!] = c[uz] = c[x]$. (Here $z = [\![Z]\!]$.)

[iiia]   Let $z = [\![Z]\!]$ and $y = [\![Y]\!]$.
Then $[\![g]\!] = c[uz + i(uz + y)] = c[x + i(x + y)]$ and $[\![g']\!] = c[i(uz + y)] = c[i(x + y)]$;
or, if Z is empty: $[\![g]\!] = c[u + i(u + y)] = c[x + i(x + y)]$ and $[\![g']\!] = c[i(x + y)]$;

or, if Y is empty, $[\![g]\!] = c[x + ix]$ and $[\![g']\!] = c[ix]$.
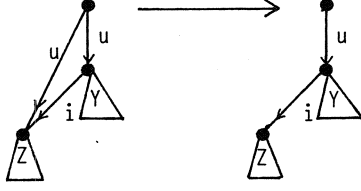
[iiib] Then $[\![g]\!] = c[uz + u(iz + y)]$ and $[\![g']\!] = c[u(iz + y)]$; or if Z is empty, $[\![g]\!] = c[u + u(i + y)]$ and $[\![g']\!] = c[u(i + y)]$.

Thus we find that $\underleftrightarrow{}_I$ on A_ω is the transitive reflexive closure of the following equation schemes:

(1) $ix \underleftrightarrow{} x$

(2) $c[xi] \underleftrightarrow{} c[x]$

(3) $c[x + x] \underleftrightarrow{} c[x]$

(4) $c[x + i(x + y)] \underleftrightarrow{} c[i(x + y)]$

(5) $c[x + ix] \underleftrightarrow{} c[ix]$

(6) $c[uz + u(iz + y)] \underleftrightarrow{} c[u(iz + y)]$

(7) $c[u + u(i + y)] \underleftrightarrow{} c[u(i + y)]$

(8) $c[i] \underleftrightarrow{} c[j]$

where $u \in E \cup I$, $i, j \in I$ and $c[\ ]$ is a $+, \cdot$-context. Note that (1) is not applied in a context. Further, (3) holds already in A_ω. Equation scheme (4) follows from (5):

$$c[x + i(x + y)] \overset{(5)}{\underleftrightarrow{}} c[x + x + y + i(x + y)] \overset{(5)}{\underleftrightarrow{}} c[x + y + i(x + y)] \underleftrightarrow{}$$
$$c[i(x + y)].$$

Equation scheme (6) needs only to be given for $u = a \in E$, because the case $u = i$ follows by taking iz for x in (4); likewise (7) follows from (6) by taking $z = i$ and applying (2). This proves the theorem. $\square$

For externally guarded terms (see 3.2) the equation I* in the preceding theorem is not necessary. Hence we arrive at the following complete axiomatization of bisimulation modulo I on guarded terms; by section 3.4 bisimulation is a congruence for such terms, even in the presence of the operators $\lfloor\!\lfloor$ and $\|$.

8.2.2. <u>COROLLARY</u>. *The congruence bisimulation modulo I on the algebra of finite, guarded processes* $G_\omega(E,I)(+,\cdot,\lfloor\!\lfloor,\|)$ *is completely axiomatized by the following equations:*

PAI:

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y).z = x.z + y.z$	A4
$(x.y).z = x.(y.z)$	A5
$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$	M1
$a \lfloor\!\lfloor x = a.x$	M2
$ax \lfloor\!\lfloor y = a(x \parallel y)$	M3
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	M4
$i = j$	I1
$xi = x$	I2
$ix + x = ix$	I3
$a(ix + y) + ax = a(ix + y)$	I4

8.2.3. <u>COROLLARY</u>. *In the initial algebra A_ω^I of* PAI, *equality of terms implies bisimulation modulo* I. *On the subalgebra of A_ω^I of guarded terms, equality coincides with bisimulation modulo* I; *in other words,* $G_\omega(E,I)(+,\cdot,\lfloor\!\lfloor,\parallel)/\underline{\leftrightarrow}_I$ *can be isomorphically embedded in* A_ω^I. $\square$

REFERENCES

[1]   ARNOLD, A.,
        Sémantique des processus communicants,
        R.A.I.R.O. Informatique théorique/Theoretical Informatics, Vol.15,
        No.2,p.103-139 (1981).

[2]   BACK, R.J.R. & H. MANNILA,
        A refinement of Kahn's semantics to handle non-determinism and communication,
        Extended abstract, Preprint University of Helsinki, 1982.

[3]   DE BAKKER, J.W. & J.I. ZUCKER,
        Denotational semantics of concurrency,
        Proc. 14th ACM Symp. on Theory of Computing,p.153-158 (1982).

[4]   DE BAKKER, J.W. & J.I. ZUCKER,
        Processes and the denotational semantics of concurrency,
        Department of Computer Science Technical Report IW 209/82,
        Mathematisch Centrum, Amsterdam 1982.

[5]   BERGSTRA, J.A. & J.W. KLOP,
        Fixed point semantics in process algebras,
        Department of Computer Science Technical Report IW 206/82,
        Mathematisch Centrum, Amsterdam 1982.

[6]   BERGSTRA, J.A. & J.W. KLOP,
        Process algebra for communication and mutual exclusion,
        Department of Computer Science Technical Report IW 218/83,
        Mathematisch Centrum, Amsterdam 1983.

[7]   BERGSTRA, J.A. & J.W. KLOP,
        A process algebra for the operational semantics of static data flow networks,
        Department of Computer Science Technical Report IW 222/83,
        Mathematisch Centrum, Amsterdam 1983.

[8]   BROCK, J.D. & W.B. ACKERMAN,
        Scenarios: a model of non-determinate computation,
        Proc. Formalization of Programming concepts (J. Díaz & I. Ramos,
        eds.),p.252-259, Springer LNCS 107, 1981.

[9]   HENNESSY, M.,
        Synchronous and asynchronous experiments on processes,
        Internal report CSR-125-82, University of Edinburgh, Department
        of Computer Science.

[10] HENNESSY, M. & R. MILNER,
        On observing nondeterminism and concurrency,
        Proc. of ICALP 80, Springer LNCS 85,p.299-309.

[11] KLOP, J.W.,
        Combinatory Reduction Systems,
        Mathematical Centre Tracts 127, Mathematisch Centrum, Amsterdam
        1980.

[12] MILNE, G.J.,
        Abstraction and nondeterminism in concurrent systems,
        Proc. FOCS, IEEE 1982 , p.358-364.

48

[13] MILNER, R.,
    A Calculus for Communicating Systems,
    Springer LNCS 92, 1980.

[14] MILNER, R.,
    A Complete Inference System for a class of Regular Behaviours,
    Preprint, Edinburgh 1982.

[15] PARK, D.M.R.,
    Concurrency and automata on finite sequences,
    Computer Science Department, University of Warwick (1981).

[16] PRATT, V.R.,
    On the composition of processes,
    Proc. 9th ACM Symp. on Principles of Programming Languages,
    p.213-223, 1982.

[17] REM, M.,
    Partially ordered computations, with applications to VLSI design,
    Proc. of the 4th Advanced Course on Foundations of Computer
    Science, Part 2, Mathematical Centre Tracts 159, Mathematisch
    Centrum 1983

69 D 13
69 F 11
69 F 12
69 F 32