Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Bergstra, J.W. Klop, E.-R. Olderog

Readies and failures in the algebra of communicating processes
(revised version)

Department of Computer Science Report CS-R8748 October

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Readies and Failures in the Algebra of Communicating Processes (revised version)

J.A. Bergstra *
Computer Science Department, University of Amsterdam ,
Kruislaan 409, 1098 SJ Amsterdam;
Department of Philosophy, State University of Utrecht,
Heidelberglaan 2, 3584 CS Utrecht, The Netherlands.

J.W. Klop *
Centre for Mathematics and Computer Science,
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands.

E.-R. Olderog
Institut für Informatik und Praktische Mathematik,
Christian-Albrechts-Universität Kiel,
2300 Kiel 1, Federal Republic of Germany.

Readiness and failure semantics are studied in the setting of ACP (Algebra of Communicating Processes). A model of process graphs modulo readiness equivalence, respectively failure equivalence, is constructed, and an equational axiom system is presented which is complete for this graph model. An explicit representation of the graph model is given, the failure model, whose elements are failure sets. Furthermore, a characterisation of failure equivalence is obtained as the maximal congruence which is consistent with trace semantics. By suitably restricting the communication format in ACP, this result is shown to carry over to subsets of Hoare's CSP and Milner's CCS. Also, the characterisation implies a full abstraction result for the failure model. In the above we restrict ourselves to finite processes without τ-steps. At the end of the paper a comment is made on the situation for infinite processes with τ-steps: notably we obtain that failure semantics is incompatible with Koomen's fair abstraction rule, a proof principle based on the notion of bisimulation. This is remarkable because a weaker version of Koomen's fair abstraction rule is consistent with (finite) failure semantics.

*) Authors partially supported by the European Communities under ESPRIT contract 432, An Integrated Formal Approach to Industrial Software Development (Meteor).
1985 Mathematics Subject Classification: 68Q05, 68Q10, 68Q55, 68Q45.
1987 CR Categories: F.1.2, F.3.1, F.3.2.
Key Words & Phrases: process algebra, concurrency, readiness semantics, failure semantics, bisimulation semantics.
Note: This paper is an extensive revision of CWI Report CS-R8523; chapters 5,6,7 have been completely rewritten and contain new material.

Introduction

This paper is concerned with the *failure semantics* for communicating processes as introduced by Brookes, Hoare and Roscoe [BHR84] (see also Rounds and Brookes [RB81].) This notion of failure semantics is based on the assumption that all possible knowledge about a process takes the form of a set of pairs $[\sigma,X]$ where σ is a linear history of events (actions) in which the process has engaged in cooperation with its environment and where X is a set of events which are impossible after σ. Thus failure semantics can be seen as a linear history semantics enriched by "local branching information".

Two further semantic models of processes will play an auxiliary role in our paper: Milner's

model based on the notion of *observational equivalence* [Mi80] or *bisimulation* (see Park [Pa83]) and the *readiness semantics* described in [OH83]. Processes which are equivalent in the sense of bisimulation semantics are also failure equivalent, but failure semantics identifies more processes. Intermediate between bisimulation and failure semantics is the readiness semantics; here positive information (σ,Y) is given about a process: Y is a set of possible actions after the history σ.

Related to the study of failure semantics which was done by Brookes, Hoare and Roscoe [BHR84] and Brookes [Br83] in the context of CSP (Communicating Sequential Processes, see [12,13]) is the work of De Nicola and Hennessy [DH84] where some equivalences, based on the notion of *test*, are introduced, one of which coincides on a class of simple expressions with failure equivalence. The work of [DH84] takes place in the context of CCS, Milner's Calculus of Communicating Systems. Connections between CCS and CSP as regards failure semantics, were given by Brookes [Br83].

Most of the work just mentioned was carried out in a context where both recursion and hiding (abstraction from silent τ-steps) were present. This combination has complicated matters significantly. The aim of our paper is therefore to investigate the "pure" failure semantics *without recursion and hiding* (except for an interesting digression in its final section where the intricate interplay of these phenomena is highlighted). Our context will be ACP, the axiomatic system for the *Algebra of Communicating Processes* as introduced and studied in the series of papers [BK83; BK84a,b,c; BK85; BBK85; BK86a,b]. (For an introductory survey see [BK86b].) As we shall see, one advantage of this choice is that the different communication concepts of CSP and CCS can be treated in a uniform way. (Cf. also Milner [Mi83] and Winskel [Wi83].) In fact, to achieve this uniformity we will work here with a mild extension of ACP where *renaming operators* are present. This system is called ACP_τ and displayed in Table 1. Note that ACP_τ is purely equational and, for a finite alphabet of actions, it is a finite axiom system.

It turns out that in our restricted setting readiness and failure semantics have a neat *axiomatisation*, by means of two equations R1,2 which on top of ACP_τ yield readiness semantics, and a "saturation" axiom S which when added to $ACP_\tau + R1,2$ yields failure semantics. ACP_τ alone corresponds to bisimulation semantics. These results are established in the first part of the paper. In Sections 1-3 we construct models for these axiom systems, starting from a domain of finite process graphs on which equivalences $\leftrightarroweq$, $\leftrightarroweq_{\mathcal{R}}$, $\leftrightarroweq_{\mathcal{F}}$ (bisimulation equivalence, readiness equivalence, failure equivalence) are divided out. Next, in Section 4, the axiom systems for these quotient structures are presented and shown to be complete. The extra axioms R1,2 and S are not new; in a form disguised by many τ's they appear already in [Br83], and they are derivable from the axioms given in [DH84] (see our comparison in Remark 7.2.3). The definitions of $\leftrightarroweq$, $\leftrightarroweq_{\mathcal{R}}$, $\leftrightarroweq_{\mathcal{F}}$ are also standard. What seems new in our treatment is the strategy of the completeness proofs by means of a decomposition of $\leftrightarroweq$, $\leftrightarroweq_{\mathcal{R}}$, $\leftrightarroweq_{\mathcal{F}}$ on process graphs in a small number of very simple *process graph transformations* (Section 3).

So we obtain a "graph model" for ACP_τ satisfying failure semantics. In Section 5, an *explicit representation* of this graph model, called the *failure model* is constructed directly from the failure sets. This links our work with that of [BHR84]. The graph model and the failure model are shown to be isomorphic. In Section 6 we restrict the general communication format of ACP_τ to 1-1

communication. We show that subsets of CSP and CCS can be interpreted within this framework. This serves as a preparation for Section 7 where we prove that for ACP_r with 1-1 communication failure equivalence is the *maximal trace respecting congruence*. Here traces are understood as *complete* histories recording all communications up to a final process state. This simple characterisation of failure equivalence seems new. In the proof we use the readiness semantics as a "stepping stone" towards failure equivalence. The characterisation is shown to carry over to the subsets of CSP and CCS introduced in Section 6. For CCS we relate our result to the notion of testing introduced in [DH84]. Further on, the characterisation implies that for ACP_r with 1-1 communication the failure model is *fully abstract* with respect to trace equivalence.

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$x + \delta = x$	A6
$\delta x = \delta$	A7
$a \mid b = b \mid a$	C1
$(a \mid b) \mid c = a \mid (b \mid c)$	C2
$\delta \mid a = \delta$	C3
$x \Vert y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	CM1
$a \mathbin{\lfloor\!\lfloor} x = ax$	CM2
$ax \mathbin{\lfloor\!\lfloor} y = a(x \Vert y)$	CM3
$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	CM4
$ax \mid b = (a \mid b)x$	CM5
$a \mid bx = (a \mid b)x$	CM6
$ax \mid by = (a \mid b)(x \Vert y)$	CM7
$(x + y) \mid z = x \mid z + y \mid z$	CM8
$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$\partial_H(xy) = \partial_H(x) \cdot \partial_H(y)$	D4
$a_H(b) = b$ if $b \notin H$	RN1
$a_H(b) = a$ if $b \in H$	RN2
$a_H(x + y) = a_H(x) + a_H(y)$	RN3
$a_H(xy) = a_H(x) \cdot a_H(y)$	RN4

ACP_r

Table 1

Algebra of Communicating Processes with renaming. Here a,b range over the set A_δ (= $A \cup \{\delta\}$) of atomic processes or actions; $\delta \notin A$ is a constant denoting deadlock; x,y,z range over the set of all processes which includes A_δ and is closed under the binary operations $+, \cdot, \Vert, \mathbin{\lfloor\!\lfloor}, \mid$ and the unary operations ∂_H, a_H where $H \subseteq A$. See Section 1.2 for further explanation.

The paper concludes in Section 8 with a digression in which processes under failure semantics are considered in the context of recursion and hiding. The main point made here is that the proof principle KFAR (*Koomen's fair abstraction rule*), which is important in system verification and which can be justified in bisimulation semantics, is not valid in any extension of (finite) failure semantics. As far as we know this observation, which is supported by deriving a formal inconsistency, is new. Remarkably, a weaker version of KFAR turns out to be both useful for verification and consistent with finite failure semantics (see [BKO86]).

Acknowledgement. We thank one of the referees for pointing out some inconsistencies in a previous version of this paper and for many detailed suggestions and corrections.

We conclude this introduction with a table of contents.

Contents

1. The domain $\mathbb{H}_\delta$ of finite acyclic process graphs

In order to build a 'graph model' for the axiomatisation ACP_r (see Introduction, Table 1) which moreover satisfies failure semantics, we start by introducing a domain of process graphs ($\mathbb{H}_\delta$) enriched with a number of operations $+,\cdot,\|,\lfloor\!\lfloor,|,\partial_H,a_H$ ($a \in A$) corresponding to the operators in ACP_r. It should be emphasized that this structure $\mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\lfloor,|,\partial_H,a_H,a,\delta)$ ($a \in A$) is not yet a model of ACP_r; it becomes so after dividing out by a suitable equivalence on $\mathbb{H}_\delta$ (which of course should be a congruence with respect to the operations). For example, dividing out by bisimulation equivalence (as defined in Section 2.3 below) yields a model of ACP_r; in fact one that is isomorphic to the initial model of ACP_r. This is however not the matter that concerns us in this paper. What we are interested in, is the quotient structure obtained by dividing out by readiness equivalence or failure equivalence respectively (defined below in 2.2): that is what we will call (in analogy with 'term model') the graph model for ACP_r, satisfying readiness semantics or failure semantics respectively.

1.1. Finite acyclic process graphs in δ-normal form.

A process graph over a set is a rooted, directed multigraph whose edges are labeled by elements of this set. Let $\mathbb{H}$ be the collection of *finite acyclic* process graphs over the alphabet $A_\delta = A \cup \{\delta\}$ (here $\delta \notin A$) consisting of actions a,b, ... $\in A$ and the constant δ denoting deadlock. We will work in the sequel with $\mathbb{H}_\delta \subseteq \mathbb{H}$, the subset of δ-normal process graphs. A process graph $g \in \mathbb{H}$ is δ-normal if whenever an edge $s \to_\delta t$ occurs in g, then the node s has outdegree 1 and the node t has outdegree 0. In anthropomorphic terminology, let us say that an edge $s \to t$ is an *ancestor* of $s' \to t'$ if it is possible to move along edges from t to s'; likewise the latter edge will be called a *descendant* of the former. Edges having the same initial node are *brothers*. So, a process graph g is δ-normal if all its δ-edges have no brothers and no descendants.

Note that for $g \in \mathbb{H}$ the ancestor relation is a partial order on the set of edges of g.

We will now associate to a process graph $g \in \mathbb{H}$ a unique g' in δ-normal form, by the following procedure:

(1) *nondeterministic δ-removal* is the elimination of a δ-edge having at least one brother,

(2) δ-*shift* of a δ-edge $s \to_\delta t$ in g consists of deleting this edge, creating a fresh node t' and adding the edge $s \to_\delta t'$.

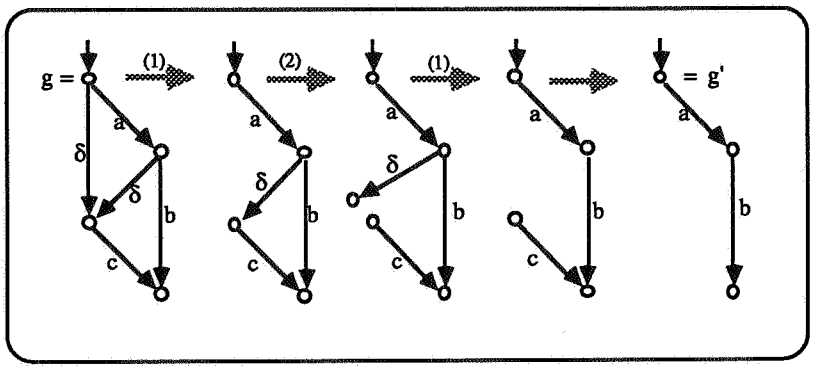


Figure 1

Now it is not hard to see that the procedure of repeatedly applying (in arbitrary order) (1),(2) in g will lead to a unique graph g' which is δ-normal; this g' is the δ-*normal form* of g. It is understood that pieces of the graph which have become unaccessible from the root, are discarded.

1.1.1. EXAMPLE. See Figure 1 where g' is the δ-normal form of g.

1.2. Operations on process graphs.

On $\mathbb{H}_\delta$ we define the operations $+,\cdot,\|,\lfloor\!\lfloor,|,\partial_H$, as in [BK85,86], and moreover renaming operators a_H. The constants a, δ (a ∈ A) are represented by graphs consisting of a single arrow labeled by a,δ respectively. For the sake of completeness we repeat the definitions briefly:

(i) the *sum* g + h is the graph obtained by identifying the roots of g,h and taking the δ-normal form (this is necessary if g or h is the graph consisting of a single step labeled with δ);

(ii) the *product* g·h is obtained by appending h at all terminal nodes which are not terminal nodes of a δ-step;

(iii) the *merge* g || h consists of the δ-normal form of the process graph obtained as the cartesian product of g,h augmented with diagonal edges for successful communications;

(iv) the *left-merge* g $\lfloor\!\lfloor$ h is the subgraph of g || h where an initial step must be one from g;

(v) the *communication merge* g || h where an initial step must be a communication result of an initial step in g and an initial step in h;

(vi) the *encapsulation* $\partial_H(g)$ is the result of renaming all (labels of) steps in H ⊆ A by δ, and taking the δ-normal form;

(vii) the *renaming* $a_H(g)$ is the result of renaming all (labels of) steps in H ⊆ A by a. We have renamings a_H for each a ∈ A.

1.2.1. EXAMPLE. Let g be the process graph in Figure 2a (next page) and h the process graph in Figure 2b. Let the communication function $|: A_\delta \times A_\delta \to A_\delta$ be such that a|c = e and b|d = f, all other communications equal δ. Then g + h is the graph in Figure 2c; g·h is the graph in 2d; g || h is the δ-normal form of the graph in 2e, which is the graph in 2f; g $\lfloor\!\lfloor$ h is the graph in 2g; g | h is the graph in 2h; $\partial_{\{a,d\}}(g)$ is the graph in 2i; $\partial_{\{a,d\}}(h)$ is the graph in 2j; and $a_{\{b\}}(g)$ is the graph in Figure 2k.

2. Equivalences on process graphs

Though in this paper our main interest is for the *ready equivalence* and *failure equivalence*, we also will consider trace equivalence and bisimulation equivalence. In this section these notions are introduced and compared. At the end of the section the concept of a *convexly saturated* process graph is introduced, which illuminates the relationship between ready and failure equivalence and which will play an important role in establishing the completeness of the axiom systems for ready and failure equivalence, respectively, presented in Section 4.

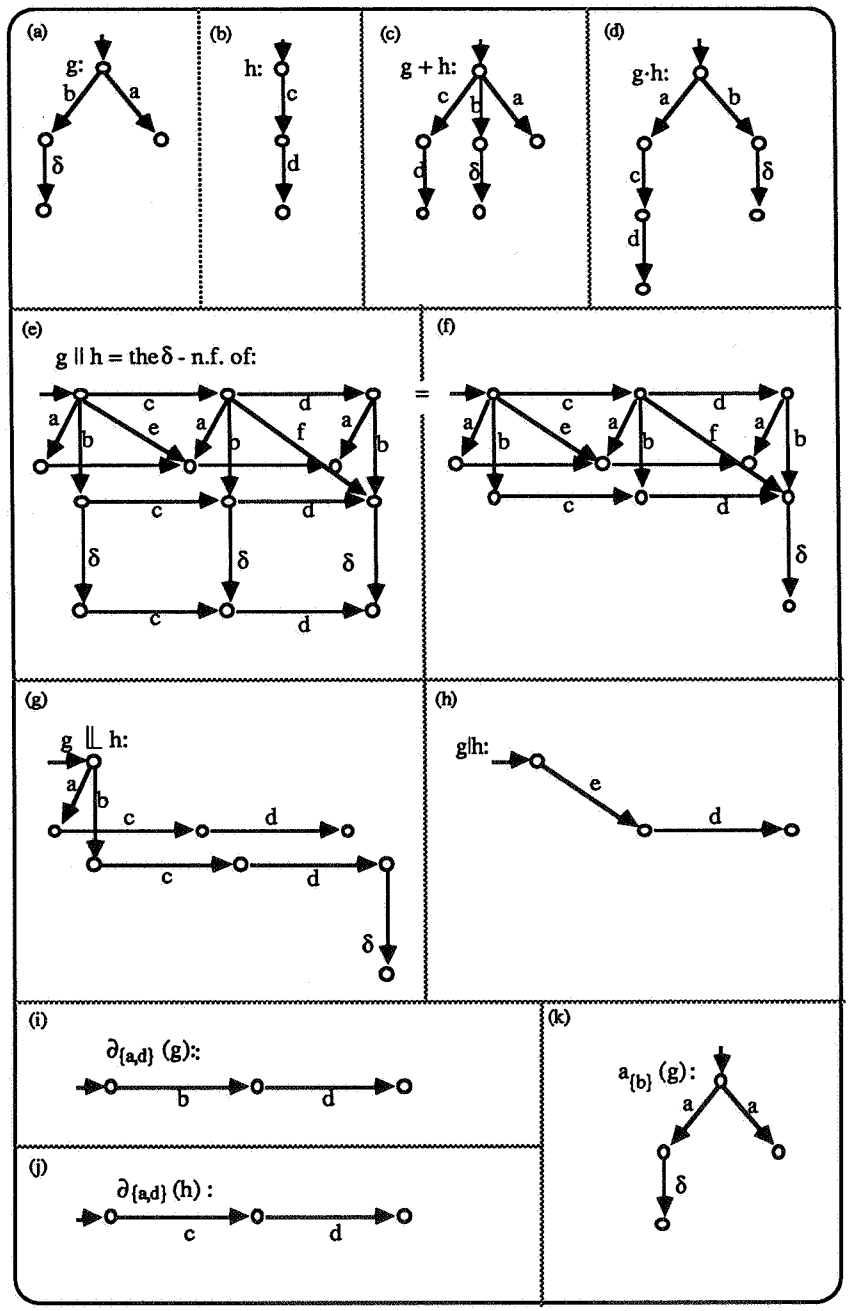


Figure 2

2.1. Trace equivalence.

Consider a process graph $g \in \mathbb{H}_\delta$. Every path in g from the root of g to some node in g determines a word $s \in A_\delta^*$ formed by concatenating the labels in the consecutive steps in the path. Any such word σ will be called a *history* of (the path in) g. We are particularly interested in *complete histories*, i.e. words determined by paths ending in a terminal node. Throughout this paper complete histories will be called *traces*. By *trace*(g) we denote the set of all traces of g. *Trace equivalence* $\sim_{tr}$ of process graphs $g,h \in \mathbb{H}_\delta$ is defined as follows:

$g \sim_{tr} h$ iff *trace*(g) = *trace*(h).

Note that there are two types of traces: *successful traces* $\sigma \in A^*$ ending in a successful termination node (see 2.2 below) and *deadlocking traces* $\sigma \cdot \delta \in A^* \cdot \{\delta\}$ ending in δ.

2.2. Ready equivalence and failure equivalence.

We will distinguish four types of nodes of $g \in \mathbb{H}_\delta$.

(i) End nodes of δ-steps in g are *improper*.

(ii) Begin nodes of δ-steps are called *deadlock nodes*.

(iii) Termination nodes of g other than those in (i) are *successful termination nodes*.

(iv) Non-terminal nodes which are not deadlock nodes.

The *successor set* of node s as in (ii) is, by definition, $\emptyset$. The successor set of a node s as in (iv) is the set of labels $\in A$ of edges with begin node s. A node as in (i) or (iii) has no successor set.

Now (σ, X) where $\sigma \in A^*$, $X \subseteq A$ is a *ready pair* of g if there is a path from root s_0 to some proper node s which is not a successful termination node, with history σ and X as the successor set of s. The *ready set* of g is the set of all ready pairs of g together with all successful traces. Notation: $\mathcal{R}[g]$.

The *failure set* of g, notation: $\mathcal{F}[g]$, is defined as follows. If $(\sigma, X) \in \mathcal{R}[g]$, then [s, Y] is a *failure pair* of g if $Y \subseteq X$, and Y is called a *refusal set*. Here and in the sequel we use the notation: $X^c = A - X$. Now $\mathcal{F}[g]$ is the set of all failure pairs of g, together (again) with the successful traces of g. Thus we have:

$$\mathcal{R}[g] = \{\sigma \mid \sigma \text{ is successful trace of g}\} \cup \{(\sigma, X) \mid (\sigma, X) \text{ is ready pair of g}\},$$
$$\mathcal{F}[g] = \{\sigma \mid \sigma \text{ is successful trace of g}\} \cup \{[\sigma, Y] \mid Y \subseteq X \text{ for some } (\sigma, X) \in \mathcal{R}[g]\}.$$

Note that δ does not appear anywhere in $\mathcal{R}[g]$ and $\mathcal{F}[g]$.

2.2.1. EXAMPLE. Consider g as in Figure 3; at each node its type (i)-(iv) is indicated. Moreover Table 2 contains the contribution of each node to the failure and ready set of g.

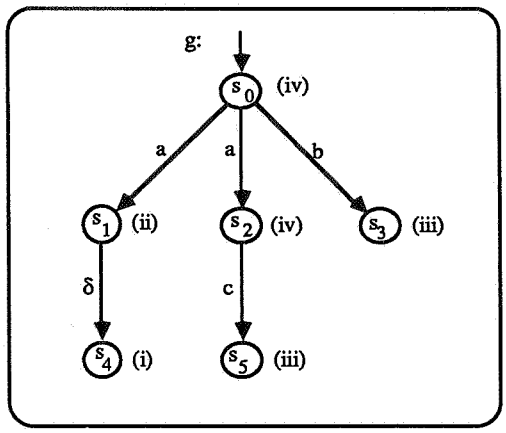


Figure 3

	$\mathcal{R}[g]$	$\mathcal{F}[g]$
s_0	$(\varepsilon, \{a,b\})$	$[\varepsilon, Y], Y \subseteq A - \{a,b\}$
s_1	$(a, \emptyset)$	$[a, Y], Y \subseteq A$
s_2	$(a, \{c\})$	$[a, Y], Y \subseteq A - \{c\}$
s_3	b	b
s_4		
s_5	ac	ac

Table 2

2.2.2. EXAMPLE.

(i) Let δ be the graph consisting of one δ-step.
Then $\mathcal{R}[\delta] = \{(\varepsilon, \emptyset)\}$ and $\mathcal{F}[\delta] = \{[\varepsilon, Y] \mid Y \subseteq A\}$.

(ii) Let $a \in A$. Then $\mathcal{R}[a] = \{(\varepsilon, \{a\}), a\}$ and $\mathcal{F}[a] = \{a\} \cup \{[\varepsilon, Y] \mid Y \subseteq A - \{a\}\}$.

(iii) Let $a\delta$ be the graph consisting of a consecutive a- and δ-step.
Then $\mathcal{R}[a\delta] = \{(\varepsilon, \{a\}), (a, \emptyset)\}$ and $\mathcal{F}[a\delta] = \{[\varepsilon, Y] \mid Y \subseteq A - \{a\}\} \cup \{[a, Z] \mid Z \subseteq A\}$.

2.2.3. DEFINITION. Let $g,h \in \mathbb{H}_\delta$. Then $g \equiv_{\mathcal{R}} h$ if $\mathcal{R}[g] = \mathcal{R}[h]$ and $g \equiv_{\mathcal{F}} h$ if $\mathcal{F}[g] = \mathcal{F}[h]$. In words: g,h are *ready equivalent*, and *failure equivalent*, respectively.

2.3. Bisimulation equivalence.

For the sake of completeness we include the definition of the well-known notion of a bisimulation.

2.3.1. DEFINITION. Let $g,h \in \mathbb{H}_\delta$. Let ROOT(g), ROOT(h) denote the root of g,h respectively and let NODES(g), NODES(h) denote the set of nodes of g,h respectively.
Then $R \subseteq$ NODES(g) $\times$ NODES(h) is a bisimulation from g to h if:

(i) (ROOT(g), ROOT(h)) $\in R$,

(ii) if $(s,t) \in R$ and $s \rightarrow_u s'$ (where $u \in A_\delta$) is an edge in g, then $(s',t') \in R$ for some t' such that $t \rightarrow_u t'$,

(iii) if $(s,t) \in R$ and $t \rightarrow_u t'$ (where $u \in A_\delta$) is an edge in h, then $(s',t') \in R$ for some s' such that $s \rightarrow_u s'$.

Notation: $g \leftrightarroweq h$ (g,h are *bisimulation equivalent*, or *bisimilar*) if there is a bisimulation from g to h (or vice versa).

As we want to model the axiom $\delta \cdot x = \delta$ later on, we profit here from the fact that only δ-normal process graphs are considered. Otherwise the definition of bisimulation would be more involved.

2.4. Comparing the equivalences. It is not hard to compare the four equivalences $\sim_{tr}$, $\equiv_{\mathcal{R}}$, $\equiv_{\mathcal{F}}$ and $\leftrightarroweq$: for $g,h \in \mathbb{H}_\delta$ we have

$$g \leftrightarroweq g \Rightarrow g \equiv_{\mathcal{R}} h \Rightarrow g \equiv_{\mathcal{F}} h \Rightarrow g \sim_{tr} h$$

and in general none of these implications can be reversed as some of the following examples (2.4.2) show. Lemma 2.5.5 states a sufficient condition for reversing the second implication.

In the sequel we will prove (Proposition 4.2.3) that $g \equiv_R h$ and $g \equiv_F h$ are *congruences* with respect to the operations defined above in 1.2. Also $\triangleq$ is a congruence; see [BK85], Theorem 2.5 for the more complicated situation where τ-steps are present. Trace equivalence however is *not* a congruence with respect to these operatrions, as the following example shows.

2.4.1. EXAMPLE. Let $\mathbb{C}[\xi]$ be the context $\partial_{\{b,c\}}(\xi \parallel c)$, and let a,b,b°,c,c° be atoms with communications b|b = b°, c|c = c° and all other communications resulting in δ. Consider the trace equivalent processes a(b + c) and ab + ac. Then $\mathbb{C}[a(b + c)] = ac° \neq ad + ac° = \mathbb{C}[ab + ac]$.

2.4.2. EXAMPLES. See Figure 4.

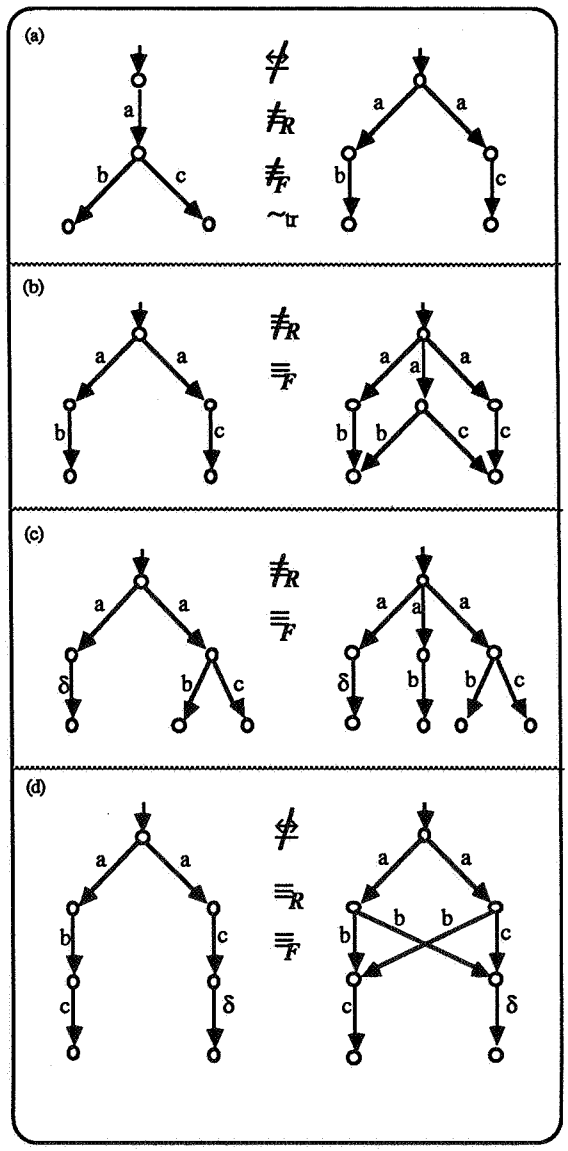


Figure 4

2.5. Convexly saturated process graphs.
Following [Br83] and [DH84]we introduce:

2.5.1. DEFINITION. $\mathcal{X} \subseteq \wp(A)$ is *convex* if
(i) $X,Y \in \mathcal{X} \Rightarrow X \cup Y \in \mathcal{X}$,
(ii) $X,Y \in \mathcal{X}, X \subseteq Z \subseteq Y \Rightarrow Z \in \mathcal{X}$.
(Here $\wp(A)$ is the power set of A. In particular, $\emptyset \subseteq \wp(A)$ is convex.)

2.5.2. DEFINITION. (i) Let $g \in \mathbb{H}_\delta$ and $\sigma \in A^*$. Then $g \mid \sigma = \{X \mid (\sigma, X) \in \mathcal{R}[g]\}$.
(ii) g is *convexly saturated* (or just 'convex' or 'saturated') if $g \mid \sigma$ is convex, for all $\sigma \in A^*$.

2.5.3. EXAMPLE. In Figure 5, g_1, g_2 are not convexly saturated, but their 'convex saturations' g_1', g_2' are.

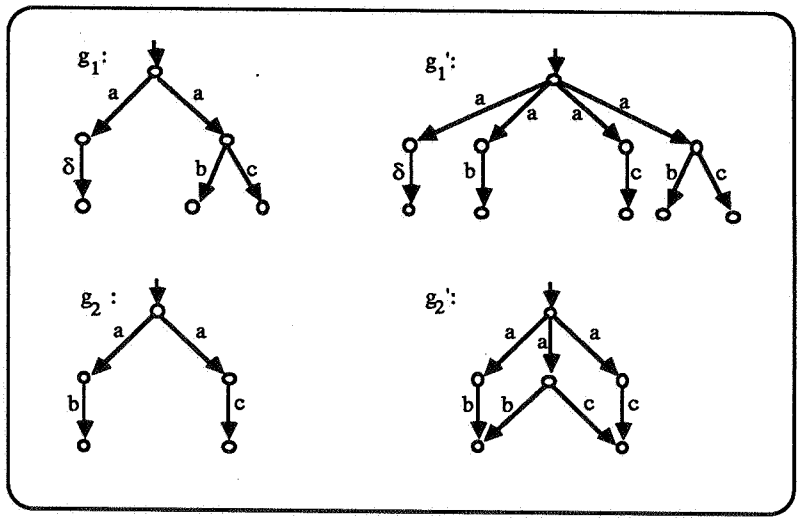


Figure 5

2.5.4. PROPOSITION. *Let* $\mathcal{X} \subseteq \wp(A)$ *be convex, and let* $Y \subseteq A$ *be a finite set such that* $Y \notin \mathcal{X}$, $Y \subseteq \bigcup\mathcal{X}$. *Then for no* $X \in \mathcal{X}$ *we have* $Y^c \subseteq X^c$.

PROOF. Consider a finite Y such that $Y \notin \mathcal{X}$, $Y \subseteq \bigcup\mathcal{X}$. Suppose that there is an $X \in \mathcal{X}$ such that $Y^c \subseteq X^c$, or equivalently $X \subseteq Y$. Clearly, Y is covered by finitely many members from $\mathcal{X}$, hence (since $\mathcal{X}$ is convex) by some $Z \in \mathcal{X}$. From $X \subseteq Y \subseteq Z$ it follows that $Y \in \mathcal{X}$, contradiction. □

2.5.5. LEMMA. *Let* $g,h \in \mathbb{H}_\delta$ *be convexly saturated. Then:*

$$g \equiv_{\mathcal{R}} h \Leftrightarrow g \equiv_{\mathcal{F}} h.$$

PROOF. Only to prove ($\Leftarrow$). So, we suppose $g \not\equiv_{\mathcal{R}} h$ and we want to prove $g \not\equiv_{\mathcal{F}} h$. We may suppose furthermore that g,h have the same trace set, otherwise $g \not\equiv_{\mathcal{F}} h$ is immediate. Now there is

a ready pair (σ, X) in (say) $\mathcal{R}[g]$ but not in $\mathcal{R}[h]$. By $(\sigma, X) \in \mathcal{R}[g]$ we have the failure pair $[\sigma, X^c] \in \mathcal{F}[g]$. Now consider $h \mid \sigma$, which is by assumption convex. Since $g \sim_{tr} h$, we have $X \subseteq \bigcup(h \mid \sigma)$. Furthermore, $(\sigma, X) \notin \mathcal{R}[h]$ entails $X \notin h \mid \sigma = \{X_i \mid i \in I\}$. So, by Proposition 2.5.4: for no $i \in I$ we have $X^c \subseteq X_i^c$. But then $[\sigma, X^c] \notin \mathcal{F}[h]$ and we have $g \not\equiv_{\mathcal{F}} h$. □

3. Transformations on process graphs

We now introduce four *elementary transformations* on process graphs $\in \mathbb{H}_\delta$ with the following property: the first two of them generate, when applied on $g \in \mathbb{H}_\delta$, all process graphs g' *bisimilar* to g; further, the first three generate the *ready equivalence* class of g; and finally, the four together generate the *failure equivalence* class of g.

3.1. The transformations double edge, sharing, cross and fork.

[i] **double edge.** This process graph transformation step removes in a *double edge* as in Figure 6 (where $a \in A$) one of the edges. Notation: $g \Rightarrow_{[i]} h$.

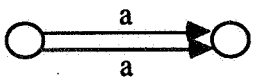


Figure 6

[ii] **sharing.** Suppose $g \in \mathbb{H}_\delta$ contains two nodes s,t determining isomorphic subgraphs $(g)_s$, $(g)_t$. Then the nodes s,t may be identified. Notation: $g \Rightarrow_{[ii]} h$.

[iii] **cross.** If $g \in \mathbb{H}_\delta$ contains a part as in Figure 7a, edges as in Figure 7b may be inserted. Notation: $g \Rightarrow_{[iii]} h$.

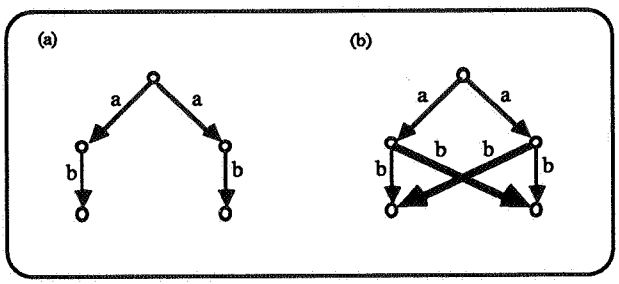


Figure 7

[iv] fork. Let $g \in \mathbb{H}_\delta$ contain a part as in Figure 8a *where all successor steps* $b_1,...,b_n$ *of the left* a-*step are displayed.* Then a part as indicated in Figure 8b may be inserted. Notation: $g \Rightarrow_{[iv]} h$.

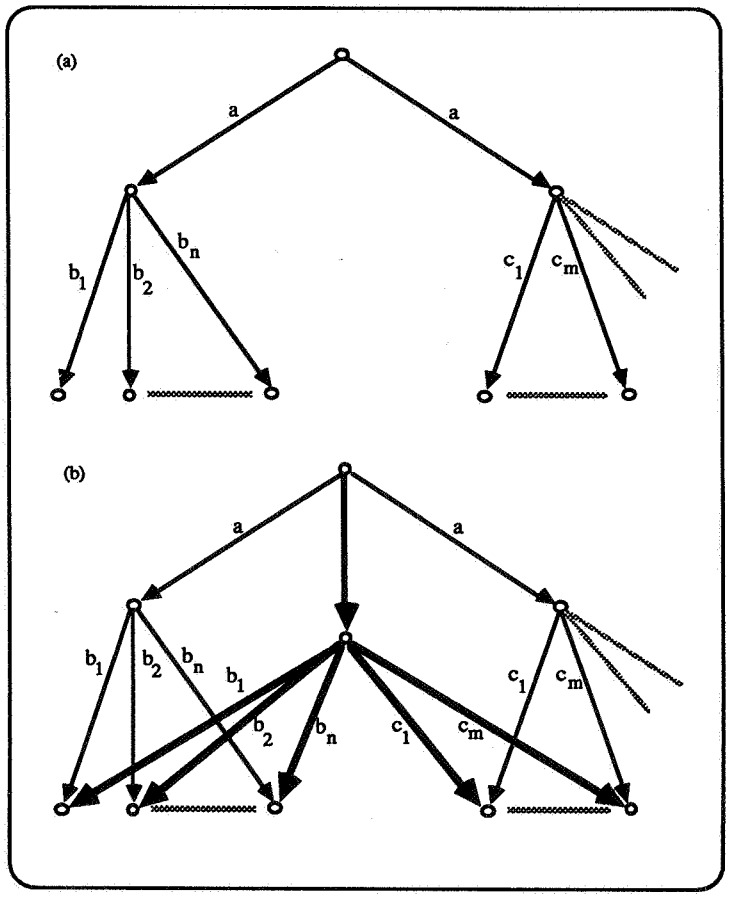


Figure 8

Here it is not required that all steps $b_1,...,b_n$, $c_1,...,c_m$ have different end nodes. If $n = 1$, b_1 may be δ; likewise c_1 may be δ. In such a case, after inserting the fork we have to δ-normalise the resulting graph again. We emphasize that a fork connects *all* of the successor steps of the left a-step with *some* of those of the right a-step.

3.1.1. NOTATION.

(i) $\Rightarrow$ is $\Rightarrow_{[i]} \cup ... \cup \Rightarrow_{[iv]}$;

(ii) $\Rightarrow^*$ is the transitive reflexive closure of $\Rightarrow$;

(iii) $\Leftrightarrow^*$ is the equivalence relation generated by $\Rightarrow$.

3.1.2. EXAMPLE. (i) See Figure 9. Note how $\Rightarrow_{[iii]}$ enables one to switch subgraphs x,y at the end of paths with the same history (abc in the following example, in Figure 9b):

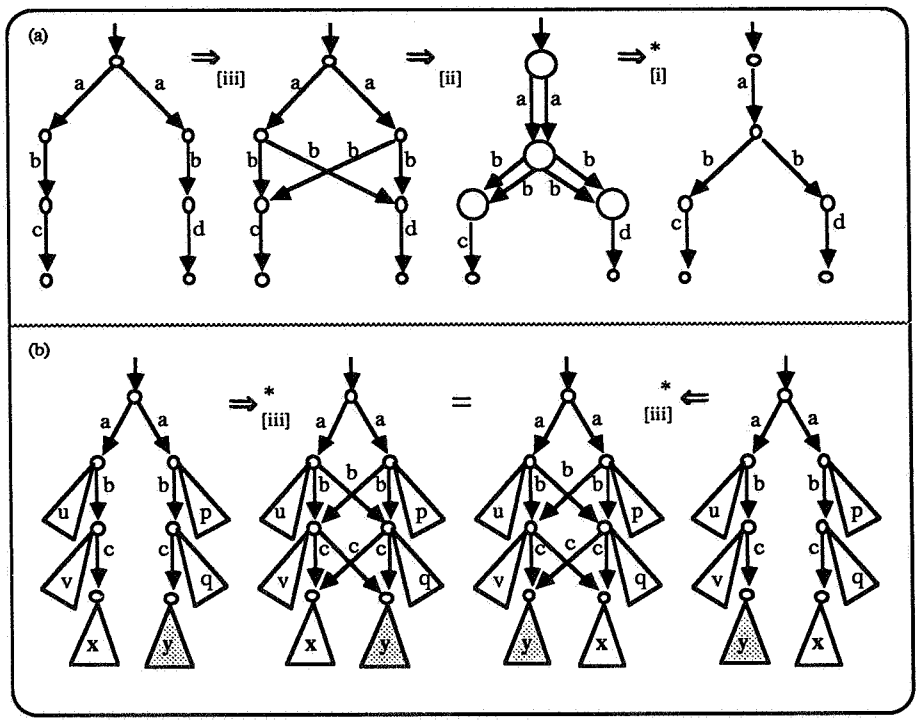


Figure 9

(ii) (See Figure 10.) Figure 10a contains an example of a fork transformation. Figure 10b contains an example of a fork transformation involving a δ-step. Figure 10c shows that complete branches can be pruned by successive transformations.

3.2. Connecting process graph equivalences with process graph transformations.

3.2.1. PROPOSITION. *Let* $g,h \in H_\delta$. *Then:*

(i) $g \Rightarrow_{[i\text{-}iii]} h$ *implies* $g \equiv_{\mathcal{R}} h$,

(ii) $g \Rightarrow_{[i\text{-}iv]} h$ *implies* $g \equiv_{\mathcal{F}} h$.

PROOF. (i) follows at once from the definitions. (ii): We must only prove that the new node s introduced in a fork does not generate new failure pairs (see Figure 8b).

Case 1. Let $(\sigma a, \{b_1,...,b_n\})$ be the ready pair contributed by node t_1, where $n \geq 1$ and the b_i are not δ. The ready pair of the new node s is $(\sigma a, \{b_1,...,b_n,c_1,...,c_m\})$. Hence the failure pairs contributed by s are among those of t_1.

Case 2. $n = 1$ and $b = \delta$. Then $(\sigma a, \emptyset)$ is the ready pair of t_1 so the failure pairs of t_1 are $[\sigma a, X]$, $X \subseteq A$ and again these cover the failure pairs of s.

Case 3. The cases where $m = 1$, $c_1 = \delta$ are trivial.

So in all cases the *new* failure pairs (of s) were already present as failure pairs of t_1. The part of $\mathcal{F}[g]$ which consists of successful traces, is invariant. □

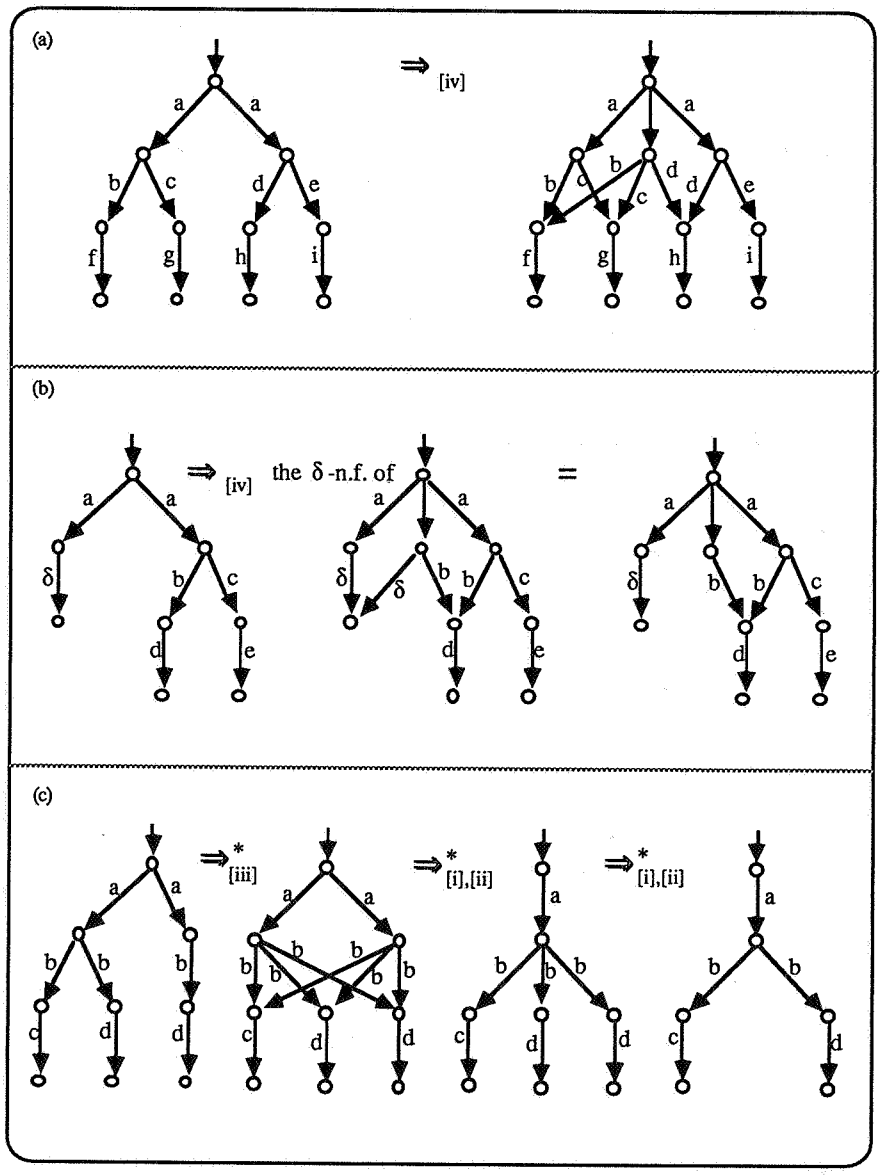


Figure 10

We will now prove the reverse implications in Proposition 3.2.1. To this end the *ready normal form* $\mathcal{R}(g)$ and the *failure normal form* $\mathcal{F}(g)$ will be defined. First we define a map γ from the collection of ready sets $\{\mathcal{R}[g] \mid g \in \mathbb{H}_\delta\}$ to $\mathbb{H}_\delta$:

3.2.2. DEFINITION. (i) Let $g \in \mathbb{H}_\delta$ have ready set $\mathcal{R}[g]$. Then $\gamma(\mathcal{R}[g])$ is the process graph with $\mathcal{R}[g] \cup \{o\}$ as set of nodes, with $(\varepsilon, X) \in \mathcal{R}[g]$ as root, and with edges given by:

$$(\sigma, \{a\} \cup X) \to_a (\sigma a, Y)$$
$$(\sigma, \{a\} \cup X) \to_a \sigma a$$
$$(\sigma, \emptyset) \to_\delta o$$

(whenever LHS, RHS $\in \mathcal{R}[g] \cup \{o\}$).

(ii) $\mathcal{R}(g) = \gamma(\mathcal{R}[g])$ is the *ready normal form* of g.

(iii) The *convex closure* $cl(\mathfrak{R}[g])$ of $\mathfrak{R}[g]$ is obtained as the smallest set containing $\mathfrak{R}[g]$ and satisfying

$$(\sigma, X), (\sigma, Y \cup Z) \in cl(\mathfrak{R}[g]) \Rightarrow (\sigma, X \cup Y) \in cl(\mathfrak{R}[g]).$$

(iv) $\mathfrak{F}(g) = \gamma(cl(\mathfrak{R}[g]))$ is the *failure normal form* of g.

3.2.3. EXAMPLE. Let g be as in Figure 11a. Then $\mathfrak{R}(g)$, $\mathfrak{F}(g)$ are as in Figure 11b, 11c respectively.

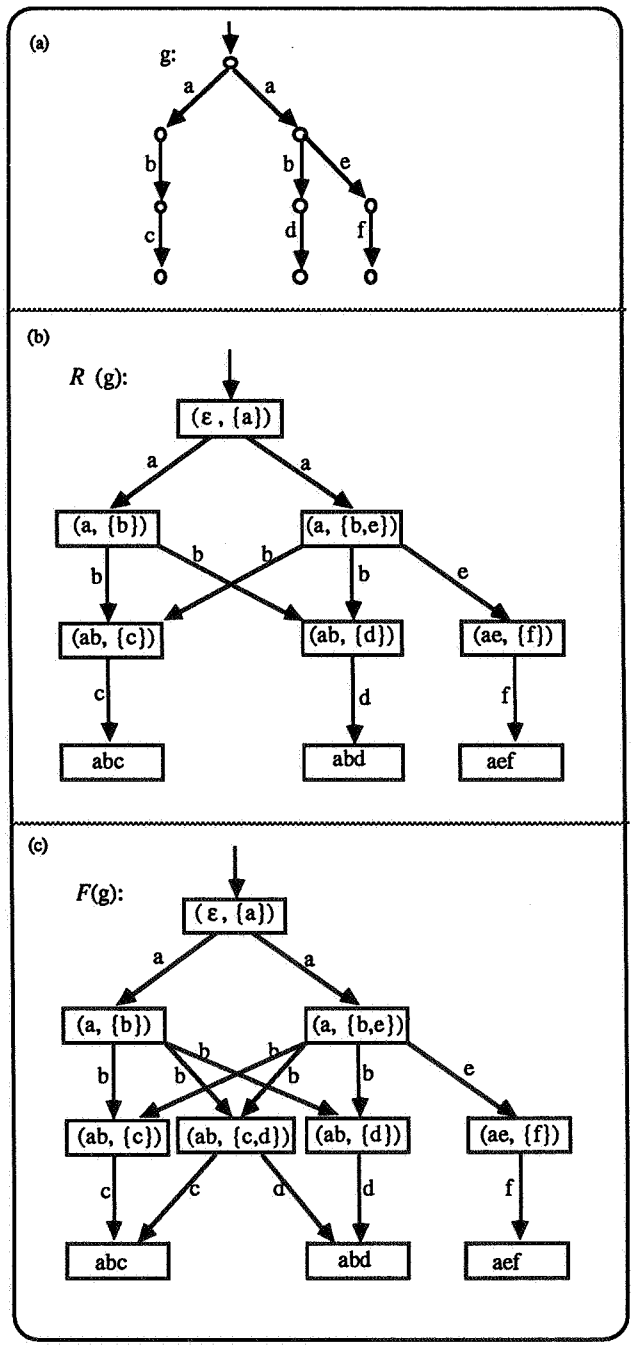


Figure 11

3.2.4. PROPOSITION.

(i) $g \Leftrightarrow^*_{[i\text{-}iii]} \mathcal{R}(g)$

(ii) $g \Leftrightarrow^* \mathcal{F}(g)$

(iii) $g \equiv_{\mathcal{R}} \mathcal{R}(g)$

(iv) $g \equiv_{\mathcal{F}} \mathcal{F}(g)$

(v) $\mathcal{R}(\mathcal{F}(g)) = \mathcal{F}(g)$

(vi) $g \equiv_{\mathcal{R}} h \Rightarrow \mathcal{R}(g) = \mathcal{R}(h)$

(vii) $g \equiv_{\mathcal{F}} h \Rightarrow \mathcal{F}(g) = \mathcal{F}(h)$.

PROOF. (i) If s is a node of $g \in \mathbb{H}_\delta$, σ is a history of s if there is a path from the root of g to s yielding the word σ. We call g *history unambiguous* if each node in g has a unique history.

Now we apply the following graph transformation procedure on $g \in \mathbb{H}_\delta$.

(1) First we make g history unambiguous by (backward) application of $\Rightarrow_{[ii]}$.

(2) Next $\Rightarrow_{[iii]}$ is applied until no further 'crosses' can be added without merely doubling edges.

(3) Then the graph is normalised with respect to $\Rightarrow_{[i]}$, $\Rightarrow_{[ii]}$. (This does not make further applications of $\Rightarrow_{[ii]}$ possible.) Call the result of the procedure (1-3): R(g).

Claim: $R(g) = \mathcal{R}(g)$.

Proof of the claim. If s is a non-terminal node of R(g), let (σ_s, X_s) be the ready pair contributed by s; if s is the terminal node of a successful trace, let σ_s be that trace. Clearly (σ_s, X_s) or σ_s, respectively, depends uniquely on s, by (1) of the procedure. Further, the ready set of g coincides with that of R(g), by Proposition 3.2.1(i).

Hence the map φ defined by $\varphi(s) = (\sigma_s, X_s)$ if s is a non-terminal node and $\varphi(s) = \sigma_s$ if s is the terminal node of a successsful trace, is in fact a map to the node set of $\mathcal{R}(g)$. It is even a bijection; for, if there were nodes s,s' in R(g) with $(\sigma_s, X_s) = (\sigma_{s'}, X_{s'})$ and $s \neq s'$, then by (2) of the construction of R(g) there are 'crosses' between each two steps a,a from s,s' respectively (see Figure 12):

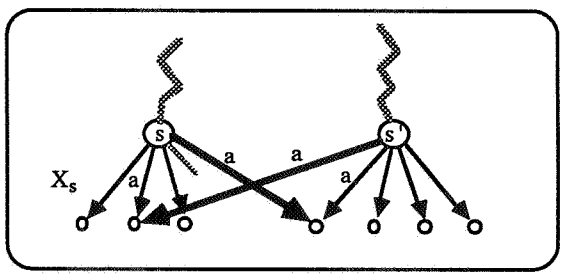


Figure 12

But this means that s,s' determine isomorphic subgraphs and are hence in stage (3) of the construction of R(g) identified; contradiction. Furthermore φ is an isomorphism:

$s \rightarrow_a t$ iff
$\varphi(s) = (\sigma_s, X_s) = (\sigma_s, \{a\} \cup X'_s)$, $\varphi(t) = (\sigma_s a, Y)$ for some X_s, X'_s, Y, iff
$\varphi(s) \rightarrow_a \varphi(t)$.

This ends the proof of the claim and thereby of part (i).

(ii) It is not hard to check that the graph which is (in the sense of γ) determined by the convex closure of $\mathcal{R}[g]$, that is $\mathcal{F}(g)$, arises from the graph $\mathcal{R}(g)$ by applying forks and crosses until modulo $\Leftrightarrow^*_{[i],[ii]}$ nothing new is added and then taking the normal form with respect to [i],[ii]. Hence it follows from (i) that $g \Leftrightarrow^* \mathcal{F}(g)$.

Parts (iii) and (iv) are left to the reader.

(v) By Definition 3.2.2, $\mathcal{R}(\mathcal{F}(g)) = \mathcal{F}(g)$ means

$$\gamma(\mathcal{R}[\gamma(cl(\mathcal{R}[g]))]) = \gamma(cl(\mathcal{R}[g])),$$

which is equivalent to

$$\mathcal{R}[\gamma(cl(\mathcal{R}[g]))] = cl(\mathcal{R}[g]).$$

So we must check that the set of ready pairs of the graph determined by the set of ready pairs $cl(\mathcal{R}[g])$ is just $cl(\mathcal{R}[g])$ and this seems obvious.

(vi) $g \equiv_{\mathcal{R}} h$ by definition means $\mathcal{R}[g] = \mathcal{R}[h]$. Hence $\mathcal{R}(g) = \gamma(\mathcal{R}[g]) = \gamma(\mathcal{R}[g]) = \mathcal{R}(h)$.

(vii) Suppose $g \equiv_{\mathcal{F}} h$. Then by (iv): $g \equiv_{\mathcal{F}} \mathcal{F}(g)$, $h \equiv_{\mathcal{F}} \mathcal{F}(h)$, so $\mathcal{F}(g) \equiv_{\mathcal{F}} \mathcal{F}(h)$. Since both $\mathcal{F}(g)$, $\mathcal{F}(h)$ are convexly closed, we have $\mathcal{F}(g) \equiv_{\mathcal{R}} \mathcal{F}(h)$ (by Lemma 2.5.5). So (vi) $\mathcal{R}(\mathcal{F}(g)) = \mathcal{R}(\mathcal{F}(h))$. Hence by (v): $\mathcal{F}(g) = \mathcal{F}(h)$. □

3.2.5. COROLLARY. *Let* $g,h \in \mathbb{H}_\delta$. *Then:*
(i) $g \triangleq h$ *iff* $g \Leftrightarrow^*_{[i],[ii]} h$
(ii) $g \equiv_{\mathcal{R}} h$ *iff* $g \Leftrightarrow^*_{[i\text{-}iii]} h$
(iii) $g \equiv_{\mathcal{F}} h$ *iff* $g \Leftrightarrow^* h$.

PROOF. (i) is (essentially) proved in [BK83] (Appendix) and also in [BK85] (Corollary 2.13): the proofs there also take τ-steps into account; after leaving out all mention of τ-steps, the result follows.
(ii) The implication from left to right follows from Proposition 3.2.1(i). The other direction follows from Proposition 3.2.4(i),(vi).
(iii) Similar to (ii). □

4. Axiomatising the equivalences on process graphs

We will now use our analysis of $\equiv_{\mathcal{R}}$, $\equiv_{\mathcal{F}}$ on the graph domain $\mathbb{H}_\delta$ to formulate complete axiom systems for these notions. First this will be done for the signature of $+, \cdot$ alone, later on (in 4.2) also $\|, \lfloor\!\lfloor, |, \partial_H$ will be taken into account.

4.1. The case without communication.

We start with the observation (whose proof is simple and omitted) that $\equiv_{\mathcal{R}}$, $\equiv_{\mathcal{F}}$ are congruences on $\mathbb{H}_\delta(+,\cdot)$ and hence can be factored out to yield $\mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{R}}$ and $\mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{F}}$ respectively. These are the structures which we will now axiomatise.

We will prove that the axiom system BPA_δ + R1,2 + S in Table 3 is a complete axiomatisation for $\mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{F}}$; after leaving out axiom S we have a complete axiomatisation for $\mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{R}}$.

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$x + \delta = x$	A6
$\delta x = \delta$	A7
$a(bx+u) + a(by+v) = a(bx+by+u) + a(bx+by+v)$	R1
$a(b+u) + a(by+v) = a(b+by+u) + a(b+by+v)$	R2
$ax + a(y + z) = ax + a(y + z) + a(x + y)$	S

BPA_δ + R1,2 + S

Table 3

Here a,b vary over $A \cup \{\delta\}$; x,y,z,u,v are variables for processes. Note that R2 is not derivable from R1 because in BPA_δ + R1,2 + S there is no process x satisfying $bx = b$, when $b \neq \delta$. On the other hand, x should be present in axiom S as the equation

$$a + a(y + z) = a + a(y + z) + ay$$

would yield the failure inconsistent equation

$$a + ab = a + ab + a\delta.$$

4.1.1. REMARK. (i) The axioms R1,2 and S (R for readiness, S for saturation) which are specific for failure equivalence, appear already in [Br83] in a slightly different form. [Br83] considers also τ-steps and presents as laws valid for failure equivalence in Proposition 1.3.6:

$$\tau(\mu x + u) + \tau(\mu y + v) = \tau(\mu x + \mu y + u) + \tau(\mu x + \mu y + v) \quad (1)$$
$$\mu x + \mu y = \mu(\tau x + \tau y) \quad (2)$$

(here $\mu \in A_\delta \cup \{\tau\}$; x,y,u,v are arbitrary processes), and in Proposition A.3 in [Br83]:

$$\tau x + \tau y = \tau x + \tau y + \tau(x + y) \quad (3)$$
$$\tau x + \tau(x + y + z) = \tau x + \tau(x + y) + \tau(x + y + z) \quad (4)$$

Clearly 1,2 imply R1 in Table 2; and using the τ-law $x\tau = x$, also valid in failure semantics, one also derives R2. Further, 3,4 together with 2 yield the pair

$$ax + ay = ax + ay + a(x + y)$$
$$ax + a(x + y + z) = ax + a(x + y) + a(x + y + z)$$

(where $a \in A_\delta$) which is equivalent to axiom S in Table 3.

(ii) The axioms R1,2 and S are also immediate consequences of the proof system of De Nicola and Hennessy [DH84] for strong testing equivalence $\simeq_2$, to be discussed and related with failure equivalence later in Remark 7.3.3. This can be seen as follows:

(1) Axiom S in Table 2: $ax + a(y + z) = ax + a(y + z) + a(x + y)$ implies

$$ax + ay = ax + ay + a(x + y)$$

by taking $z = y$; this is (D5) in [DH84]. Further, (S) implies

$$ax + a(x + y + z) = ax + a(x + y + z) + a(x + y)$$

by replacing y in (S) by $x + y$. This is (D6) in [DH84]. Vice versa, (S) follows from (D5,6):

$$ax + a(y + z) = \quad (D5)$$
$$ax + a(y + z) + a(x + y + z) = \quad (D6)$$
$$ax + a(y + z) + a(x + y + z) + a(x + y) = \quad (D5)$$
$$ax + a(y + z) + a(x + y).$$

(2) Axiom R1: $a(bx+u) + a(by+v) = a(bx+by+v) + a(bx+by+u)$ is derived from the axiom system in [DH84] as follows.

$$bx + \tau(by + v) = \tau(bx + by + v) \quad (N3)$$
$$by + \tau(bx + u) = \tau(bx + by + u) \quad (N3)$$
$$bx + by + \tau(by + v) + \tau(bx + u) = \tau(bx + by + v) + \tau(bx + by + u)$$
$$bx + \tau(bx + u) = \tau(bx + u) \quad (D9)$$

$$by + \tau(by + v) = \tau(by + v) \qquad (D9)$$
$$\tau(by + v) + \tau(bx + u) = \tau(bx + by + v) + \tau(bx + by + u)$$
$$a[\tau(by + v) + \tau(bx + u)] = a[\tau(bx + by + v) + \tau(bx + by + u)]$$
$$a(by + v) + a(bx + u) = a(bx + by + v) + a(bx + by + u) \qquad (N1)$$

Here N1,3 and D9 are axioms in [DH84].

(3) Axiom R2: $a(b + u) + a(by + v) = a(b + by + u) + a(b + by + v)$ is not needed in [DH84] because a process b which first performs action b and then successfully terminates is not considered there. Note that the process bNIL of [DH84] corresponds to $b\cdot\delta$ and is thus different from b.

4.1.2. Connecting terms with process graphs.

Let $Ter(BPA_\delta)$ be the set of closed terms in the signature of BPA_δ (= the signature of BPA_δ+ R1,2 + S). We define the following translations:

$$graph: \quad Ter(BPA_\delta) \to \mathbb{H}_\delta$$
$$ter: \quad \mathbb{H}_\delta \to Ter(BPA_\delta).$$

Here *graph*(T) is the process graph obtained by first normalizing T with respect to A4,6,7 in Table 2 and second interpreting a, +, · as the corresponding 'one edge graphs' and operators +, · on $\mathbb{H}_\delta$.

Further, to define *ter*(g) we first define *tree*(g) as the tree obtained from g by 'unsharing'. Now we define *ter*(g) as the term corresponding in the obvious way to *tree*(g).

4.1.2.1. EXAMPLE. (i) *graph*(a(b + c + d)d + de + ed) = *graph* (a(bd + cd) + ed) is the graph in Figure 13a.

(ii) If g is as in Figure 13b, then *tree*(g) is as in Figure 13c.

(iii) If g is as in (ii), then *ter*(g) = ace + b(de + ab).

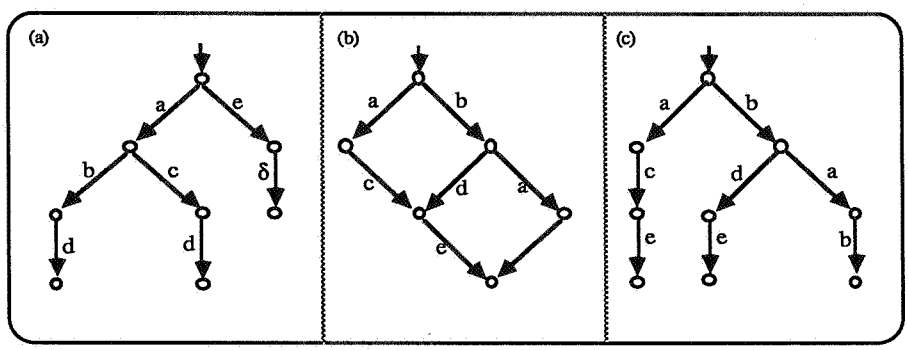


Figure 13

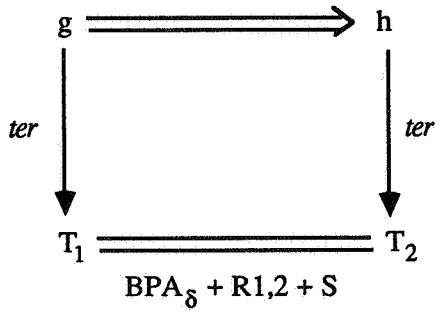
4.1.3. REMARK. Note that *ter*, *graph* are 'almost' inverse to each other:

$$BPA_\delta \vdash (ter \circ graph)(T) = T$$
$$(graph \circ ter)(g) \leftrightarrows g$$

where $\leftrightarrows$ (bisimilarity) coincides with $\Leftrightarrow^*_{[i],[ii]}$.

4.1.4. TRANSFER LEMMA. (See diagram.) *Let* $g,h \in \mathbb{H}_\delta$ *be such that* $g \Rightarrow h$. *Then*

$$BPA_\delta + R1,2 + S \vdash ter(g) = ter(h).$$



PROOF. A transformation $g \Rightarrow_{[i]} h$ (removing a double edge) 'translates' into an application of A3: $x + x = x$.

A transformation $g \Rightarrow_{[ii]} h$ is invisible on the level of terms, i.e. *ter*(g) and *ter*(h) are identical terms. Next consider a transformation $g \Rightarrow_{[iii]} h$, which consists of adding two edges in g as in Figure 14.

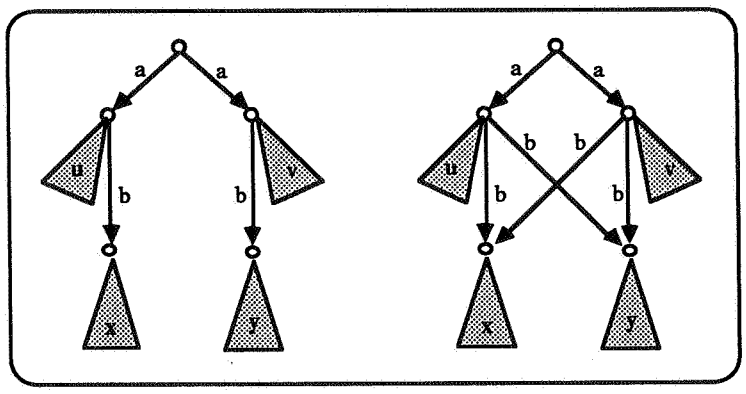


Figure 14

This translates to an application of R1 if the subtrees x,y are non-empty, and to R2 if one of these subtrees is empty. In case both subtrees x,y are empty we have an application of axiom A3.

Finally, a transformation $g \Rightarrow_{[iv]} h$ (see also Figure 8) translates into an application of axiom S in Table 2. □

4.1.5. THEOREM. (i) $BPA_\delta + R1,2 \vdash T_1 = T_2 \Leftrightarrow graph(T_1) \equiv_{\mathcal{R}} graph(T_2)$.
(ii) $BPA_\delta + R1,2 + S \vdash T_1 = T_2 \Leftrightarrow graph(T_1) \equiv_{\mathcal{F}} graph(T_2)$.

PROOF. We prove (ii); the proof of (i) is similar, noting that the proof of Lemma 4.1.4 shows that $\Rightarrow_{[i\text{-}iii]}$ is transferred to applications of axioms in $BPA_\delta + R1,2$.

Checking the soundness ($\Rightarrow$) is routine and will not be done here. As to the completeness ($\Leftarrow$): suppose $graph(T_1) \equiv_{\mathcal{F}} graph(T_2)$. Then by Corollary 3.2.5: $graph(T_1) \leftrightarrow^* graph(T_2)$. Now by the Transfer Lemma 4.1.4 we have

$$BPA_\delta + R1,2 + S \vdash (ter \circ graph)(T_1) = (ter \circ graph)(T_2)$$

and by Remark 4.1.3(ii):

$$BPA_\delta + R1,2 + S \vdash T_1 = T_2. \quad \square$$

4.1.6. NOTATION. (i) If (Σ, E) is a specification (sometimes only written as E if the signature Σ is clear), then $I(\Sigma, E)$ is its initial algebra.
(ii) $\cong$ denotes isomorphism between algebras.

4.1.7. COROLLARY. (i) $\mathbb{H}_\delta(+,\cdot,a,\delta)/\equiv_{\mathcal{R}} \cong I(BPA_\delta + R1,2)$
(ii) $\mathbb{H}_\delta(+,\cdot,a,\delta)/\equiv_{\mathcal{F}} \cong I(BPA_\delta + R1,2 + S)$. □

4.2. The case with communication: the graph model of ACP_r.

Finally we will prove the results above in the presence of communication. The operators $\|, \lfloor\!\lfloor, \cdot, l, \partial_H, a_H$ ($a \in A$) on $\mathbb{H}_\delta$ were already introduced in Section 1.2. They are the semantical counterparts of the same operators in the axiom system ACP_r, as in the upper part of Table 4, which presents the axiom system $ACP_r + R1,2 + S$, and which extends our earlier axiom system $BPA_\delta + R1,2 + S$ in Table 3.

As before, in Table 4 a,b,c vary over $A \cup \{\delta\}$, and x,y,z,u,v vary over processes.

We want to prove that the initial algebra of $ACP_r + R1,2 + S$ is isomorphic to the model of finite acyclic graphs modulo failure equivalence $\equiv_{\mathcal{F}}$, called the *graph model* for $ACP_r + R1,2 + S$. To this end we have first to prove that is a *congruence* with respect to also the new operators. Once we have this, and knowing from [BK85,86a] (after leaving out all reference to τ-steps) that there is the isomorphism

$$I(ACP_r) \cong \mathbb{H}_\delta(+,\cdot, \|, \lfloor\!\lfloor, \cdot, l, \partial_H, a_H, a, \delta)/\leftrightarrow$$

where $\leftrightarrow$ is bisimulation (which coincides with $\leftrightarrow^*_{[i],[ii]}$; Corollary 3.2.5(i)), the derived

isomorphism is a consequence from some general facts which we will state now.

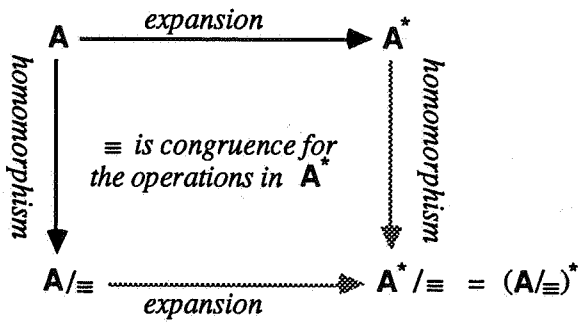
$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$x + \delta = x$	A6
$\delta x = \delta$	A7
$a \mid b = b \mid a$	C1
$(a \mid b) \mid c = a \mid (b \mid c)$	C2
$\delta \mid a = \delta$	C3
$x \Vert y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	CM1
$a \mathbin{\lfloor\!\lfloor} x = ax$	CM2
$ax \mathbin{\lfloor\!\lfloor} y = a(x \Vert y)$	CM3
$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	CM4
$ax \mid b = (a \mid b)x$	CM5
$a \mid bx = (a \mid b)x$	CM6
$ax \mid by = (a \mid b)(x \Vert y)$	CM7
$(x + y) \mid z = x \mid z + y \mid z$	CM8
$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$\partial_H(xy) = \partial_H(x) \cdot \partial_H(y)$	D4
$a_H(b) = b$ if $b \notin H$	RN1
$a_H(b) = a$ if $b \in H$	RN2
$a_H(x + y) = a_H(x) + a_H(y)$	RN3
$a_H(xy) = a_H(x) \cdot a_H(y)$	RN4
$a(bx+u) + a(by+v) = a(bx+by+u) + a(bx+by+v)$	R1
$a(b+u) + a(by+v) = a(b+by+u) + a(b+by+v)$	R2
$ax + a(y + z) = ax + a(y + z) + a(x + y)$	S

$$\mathbf{ACP_r + R1,2 + S}$$

Table 4

4.2.1. General intermezzo.

Let A be an algebra which on the one hand can be expanded to A* (i.e. enriched with new functions; the domain is invariant) and on the other hand can be factored out via $\equiv$, a congruence on A, to A/$\equiv$. Suppose moreover that $\equiv$ is also a congruence on A*. (See diagram, next page.)



Then this expansion and factorisation are compatible (or commuting): $A^*/\equiv$ equals $(A/\equiv)^*$. Now let A, A^*, $A/\equiv$ (as in I) be isomorphic respectively to the initial algebras of the equational specifications (Σ, E), $(\Sigma \cup \Delta, E \cup D)$, $(\Sigma, E \cup F)$. Then it follows that $(\Sigma \cup \Delta, E \cup D)$ is
(1) a conservative extension of the 'base' specification (Σ, E) (i.e. no new identities between closed terms in the base signature Σ are provable from $(\Sigma \cup \Delta, E \cup D)$), and
(2) moreover the extra operators in Δ can be *eliminated.*

$$(\Sigma, E) \xrightarrow{\text{conservative extension with elimination property}} (\Sigma \cup \Delta, E \cup D)$$
$$\downarrow$$
$$(\Sigma, E \cup F)$$

III. Furthermore (and this is what we are interested in) we may conclude from the given isomorphisms that

$$A^*/\equiv \;=\; (A/\equiv)^* \;\equiv\; I(\Sigma \cup \Delta, E \cup D \cup F)$$

where the last algebra is the initial algebra of the *union* of $(\Sigma, E \cup F)$ and $(\Sigma \cup \Delta, E \cup D)$.

In the statement of the next theorem, as well as in its proof and Table 5, we have suppressed mention of the constants a,δ in e.g. $\mathbb{H}_\delta(+,\cdot)$, which actually should read $\mathbb{H}_\delta(+,\cdot,a,\delta)$ $(a \in A)$.)

4.2.2. THEOREM. *Let the initial algebras* $I(BPA_\delta)$ *etc. as in Table 5(ii) of the axiom systems* BPA_δ *etc. as in Table 5(i) be given. Furthermore, consider the graph models* $\mathbb{H}_\delta(+,\cdot)/\leftrightarrows$ *etc.* as *in Table 5(iii).*

Then corresponding initial models and graph models are isomorphic. In particular:

$$I(ACP_\tau + R1,2 + S) \;\equiv\; \mathbb{H}_\delta(+,\cdot,\|,\between,\between,\partial_H,a_H)/\equiv_F.$$

(i)

$$\begin{array}{ccc} BPA_\delta & \longrightarrow & ACP_r \\ \downarrow & & \downarrow \\ BPA_\delta + R1,2 & \longrightarrow & ACP_r + R1,2 \\ \downarrow & & \downarrow \\ BPA_\delta + R1,2 + S & \longrightarrow & ACP_r + R1,2 + S \end{array}$$

(ii)

$$\begin{array}{ccc} I(BPA_\delta) & \xrightarrow{exp} & I(ACP_r) \\ \downarrow hom & & \downarrow hom \\ I(BPA_\delta + R1,2) & \xrightarrow{exp} & I(ACP_r + R1,2) \\ \downarrow hom & & \downarrow hom \\ I(BPA_\delta + R1,2 + S) & \xrightarrow{exp} & I(ACP_r + R1,2 + S) \end{array}$$

(iii)

$$\begin{array}{ccc} \mathbb{H}_\delta(+,\cdot)/\triangleq & \xrightarrow{exp} & \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\rfloor,\mathrm{I},\partial_H,a_H)/\triangleq \\ \downarrow hom & & \downarrow hom \\ \mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{R}} & \xrightarrow{exp} & \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\rfloor,\mathrm{I},\partial_H,a_H)/\equiv_{\mathcal{R}} \\ \downarrow hom & & \downarrow hom \\ \mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{F}} & \xrightarrow{exp} & \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\rfloor,\mathrm{I},\partial_H,a_H)/\equiv_{\mathcal{F}} \end{array}$$

Table 5

PROOF. Consider e.g.

$$\begin{array}{c} BPA_\delta \rightarrow ACP_r \\ \downarrow \\ BPA_\delta + R1,2 + S \end{array}$$

and the corresponding initial algebras

$$\begin{array}{c} I(BPA_\delta) \rightarrow I(ACP_r) \\ \downarrow \\ I(BPA_\delta + R1,2 + S) \end{array}$$

and furthermore the (by position in the diagram in Table 5) corresponding graph models

$$\begin{array}{ccc} \mathbb{H}_\delta(+,\cdot)/\triangleq & \xrightarrow{exp} & \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\rfloor,\mathrm{I},\partial_H,a_H)/\triangleq \\ \downarrow hom & & \\ \mathbb{H}_\delta(+,\cdot)/\equiv_{\mathcal{F}} & & \end{array}$$

By Corollary 4.1.6(ii) we have $I(BPA_\delta + R1,2 + S) \cong \mathbb{H}_\delta(+,\cdot)/\equiv_\mathcal{F}$, and by results in [BK85,86] we have $I(BPA_\delta) \cong \mathbb{H}_\delta(+,\cdot)/\leftrightarrows$ and $I(ACP_r) \cong \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\lfloor,\lfloor,\partial_H,a_H)/\leftrightarrows$.

Therefore, by 4.2.1(III), it suffices to prove that $\equiv_\mathcal{F}$ is a congruence with respect to the 'new' operators on $\mathbb{H}_\delta$ in order to conclude that

$$I(ACP_r + R1,2 + S) \cong \mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\lfloor,\lfloor,\partial_H,a_H)/\equiv_\mathcal{F}.$$

This is proved in the next proposition. □

4.2.3. PROPOSITION. (i) *Failure equivalence is a congruence with respect to the operators* $\|,\lfloor\!\lfloor,\lfloor,\partial_H,a_H$ *on* $\mathbb{H}_\delta$.
(ii) *The same holds for ready equivalence.*

PROOF. (i) We consider some typical cases.
The case of ∂_H. To prove: $g \equiv_\mathcal{F} h \Rightarrow \partial_H(g) \equiv_\mathcal{F} \partial_H(h)$. By Corollary 3.2.5 it suffices to check that $g \Rightarrow h$ implies $\partial_H(g) \equiv_\mathcal{F} \partial_H(h)$. The cases that $\Rightarrow$ is $\Rightarrow_{[i]}$ or $\Rightarrow_{[ii]}$ present no problem. As to $\Rightarrow_{[iii]}$: it is easy to verify that

$$g \Rightarrow_{[iii]} h \text{ implies } \partial_H(g) = \partial_H(h) \text{ or } \partial_H(g) \Rightarrow_{[iii]} \partial_H(h).$$

As to $\Rightarrow_{[iii]}$: as in the previous case, the effect of ∂_H (renaming some atoms in g,h into δ and δ-normalising the resulting graphs again) is such that either the 'same' fork can be inserted or $\partial_H(g) = \partial_H(h)$.

(Note here that it is crucial that process graphs g,h as in Figure 15 are not failure quivalent, since $\partial_{\{b\}}$ would yield a trace $a\delta$ in h but not in g.)

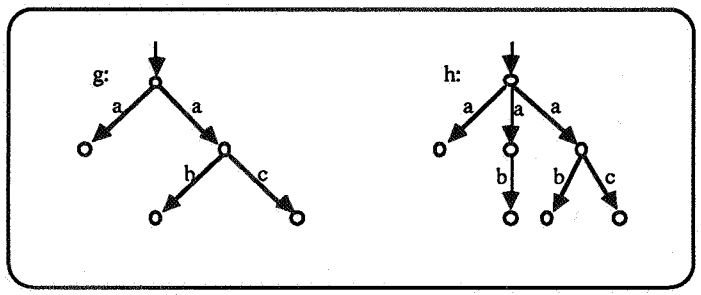


Figure 15

The case of $\|$. It suffices to prove:

$$g \Rightarrow g' \text{ implies } g \| h \equiv_\mathcal{F} g' \| h.$$

As above, only the cases [iii], [iv] (cross and fork, respectively) are of interest. In fact we will prove:

(1) $g \Rightarrow_{[iii]} g'$ implies $g \parallel h \Rightarrow_{[iii]} g' \parallel h$.
(2) $g \Rightarrow_{[iv]} g'$ implies $g \parallel h \equiv_{\mathcal{F}} g' \parallel h$.

Proof of (1): Due to the construction of a merge as a cartesian product with diagonal edges for communications (Figure 16), it is 'geometrically' clear (see Figure 17) that inserting a cross in g amounts to inserting several crosses (also possibly diagonal ones, depending on the communication function) in the merge $g \parallel h$. So $g \parallel h \Rightarrow_{[iii]} g' \parallel h$.

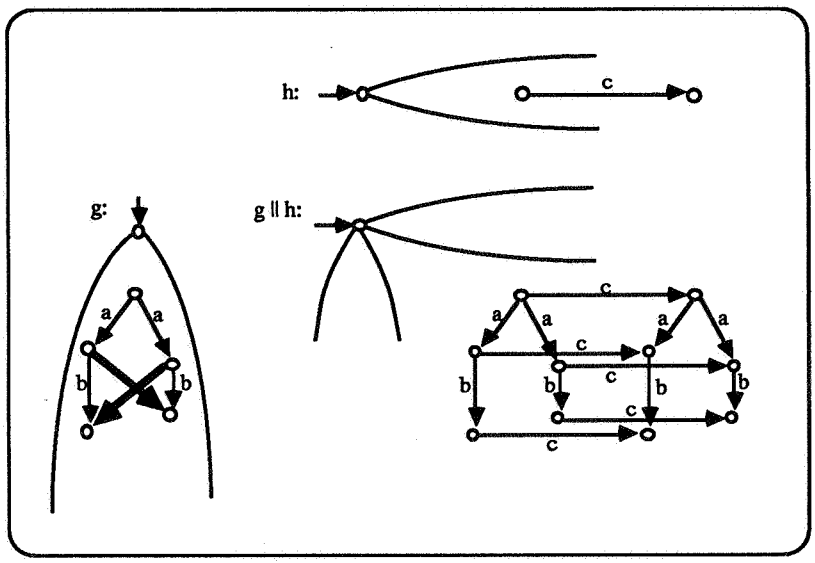


Figure 16

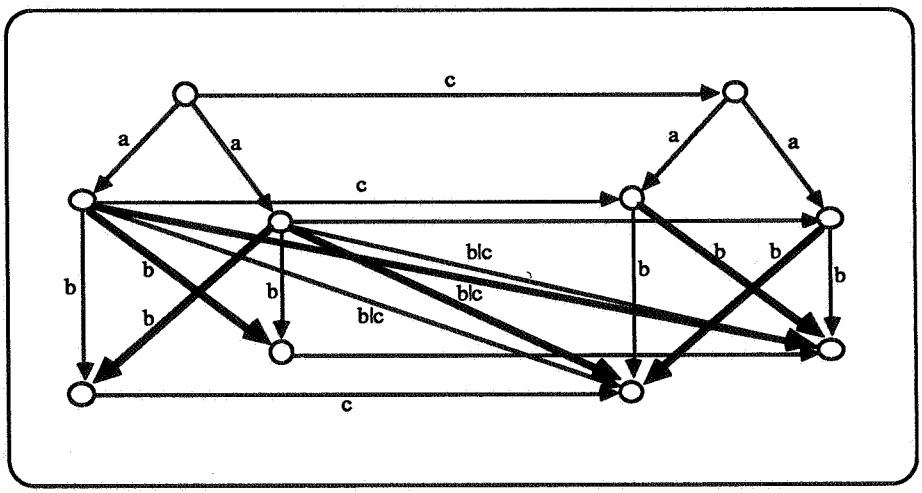


Figure 17

Proof of (2). Under the assumption $g \Rightarrow_{[iv]} g'$ we now prove $g \parallel h \equiv_F g' \parallel h$ directly from the definition of $\equiv_F$. So consider the addition in g of a fork which connects all successors of s_1 (see Figure 18) to some of those of s_3. I.e. the failure pairs contributed by the new node s_2 are contained in those of s_1. Then we must check that the new nodes (s_2,t) in $g' \parallel h$ caused by this addition, contribute no new failure pairs. It is not hard to check that indeed the failure pairs of (s_2,t) are contained in those of (s_2, t) by some consideration of the outgoing edges of (s_1,t) and (s_2,t). The precise verification is omitted here.

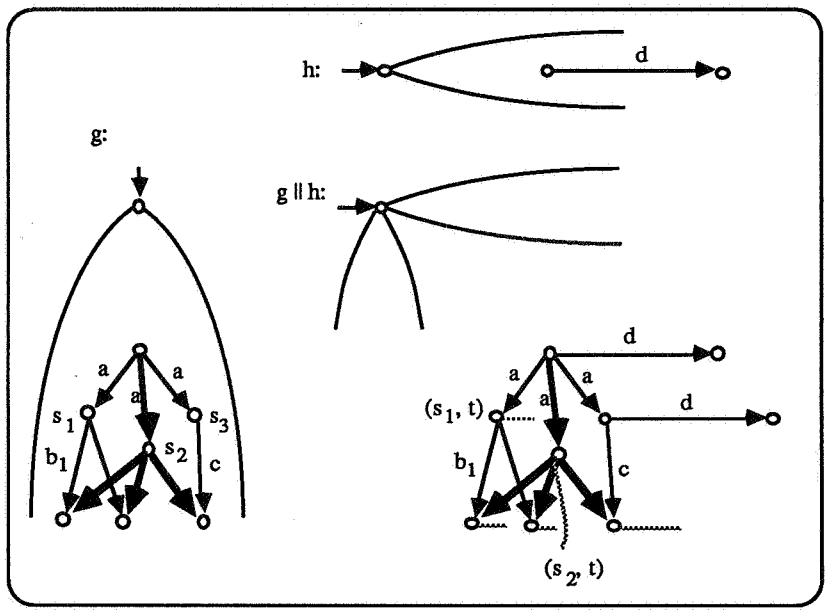


Figure 18

The proof of part (ii) of the proposition is as for (i)—but simpler. It is omitted here. □

5. The failure model of ACP_r

In the previous sections the notion of failure equivalence was introduced for the process graph domain $\mathbb{H}_\delta$, and it was shown to be a congruence with respect to the operators of ACP_r in $\mathbb{H}_\delta$. The quotient $\mathbb{H}_\delta / \equiv_F$ was shown to be a model of ACP_r, called the graph model of ACP_r. Furthermore, a complete axiomatisation $ACP_r + R1,2 + S$ was given for $\equiv_F$ in the sense of

$$I(ACP_r + R1,2 + S) \cong \mathbb{H}_\delta / \equiv_F.$$

Here $\mathbb{H}_\delta / \equiv_F$ is short for $\mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\lfloor,\mid,\partial_H,a_H,a,\delta) / \equiv_F$. In this section we will provide an *explicit representation* of the quotient structure $\mathbb{H}_\delta(+,\cdot,\|,\lfloor\!\lfloor,\mid,\partial_H,a_H,a,\delta) / \equiv_F$, called the *failure model* of ACP_r. The model will shed more light into the structure of failures, and—in connection with Section 6.2— it will link our definitions with the original work on failures in [BHR84].

5.1. The domain $\mathbb{F}$ of failure sets.

First we introduce the domain of failure sets, denoted by $\mathbb{F}$. It consists of all finite subsets

$$F \subseteq A^+ \cup (A^* \times \wp(A))$$

(where A^* is the set of finite words over A, A^+ is the set of non-empty finite words over A and $\wp(A)$ is the power set of A) which satisfy the following closure properties:

(i) $[\varepsilon, \emptyset] \in F$,
(ii) $[\sigma_1\sigma_2, \emptyset] \in F \Rightarrow [\sigma_1, \emptyset] \in F$,
(iii) $X \subseteq Y \,\&\, [\sigma, Y] \in F \Rightarrow [\sigma, X] \in F$,
(iv) $[\sigma,X] \in F \,\&\, [\sigma, X \cup \{a\}] \notin F \Rightarrow \sigma a \in F$ or $[\sigma a, \emptyset] \in F$,
(v) $\sigma a \in F \Rightarrow [\sigma, \emptyset] \in F$.

For failure sets $F \subseteq A^* \times \wp(A)$ not involving any traces $\sigma \in A^+$ these are exactly the closure properties postulated in [9]. Our reasons for allowing also (successful, non-empty) traces σ to appear in failure sets F is that they allow a direct definition of sequential composition without using (and later hiding again) an extra action $\surd$ coding the event of successful termination as in [BHR84 For processes where successful termination is possible only *after* some action $a \in A$ has occurred (as in ACP_r), the failure domain $\mathbb{F}$ is isomorphic to the one in [BHR84]. However, we will not make use of this isomorphism because in Section 6.2 on CSP we will restrict ourselves to CSP processes without successful termination.

5.2. Operations on failure sets.

Now we define the constants δ, a ($a \in A$) and the operations $+,\cdot,\|,\lfloor\!\lfloor,\partial_H,a_H$ of ACP_r directly on $\mathbb{F}$. For $F,G \in F$ we put

(i) $\delta = \{[\varepsilon, X] \mid X \subseteq A\}$.

(ii) $a = \{[a, X] \mid X \subseteq A\text{-}\{a\}\} \cup \{a\}$.
Initially 'a' can refuse anything except 'a'. After 'a' has occurred, the process successfully terminates.

(iii) $F + G = \{[\varepsilon, X] \mid [\varepsilon, X] \in F \cap G\}$
$\cup \{\sigma \mid \sigma \in F \cup G\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge [\sigma, X] \in F \cup G\}$.
In its first step F + G can refuse only those actions which can be refused by both F and G. In all subsequent steps F + G behaves like $F \cup G$.

(iv) $F \cdot G = \{[\sigma, X] \mid [\sigma, X] \in F\}$
$\cup \{\sigma_1\sigma_2 \mid \sigma_1 \in F \wedge \sigma_2 \in G\}$
$\cup \{[\sigma_1\sigma_2, X] \mid \sigma_1 \in F \wedge [\sigma_2, X] \in G\}$.
$F \cdot G$ first behaves like F and after successful termination of F in a trace σ_1 continues to behave like G.

(v) $F \parallel G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 \| \sigma_2\}$ (1)
$\cup \{[\sigma, X] \mid \exists [\sigma_1, X_1] \in F, [\sigma_2, X_2] \in G:$
$\sigma \in \sigma_1 \| \sigma_2$
$\wedge X \subseteq (X_1 \cap X_2) - \{(a|b) \mid a \notin X_1 \wedge b \notin X_2\}\}$ (2)
$\cup \{[\sigma, X] \mid \exists \sigma_1 \in F, [\sigma_2, X_2] \in G: \sigma \in \sigma_1 \| \sigma_2 \wedge X = X_2\}$ (3)
$\cup \{[\sigma, X] \mid \exists [\sigma_1, X_1] \in F, \sigma_2 \in G: \sigma \in \sigma_1 \| \sigma_2 \wedge X = X_1\}$ (4)

where $\sigma_1 \| \sigma_2$ is the set of traces in A* defined inductively by:

$\varepsilon \parallel \sigma = \sigma \parallel \varepsilon = \{\sigma\}$
$a\sigma_1 \parallel b\sigma_2 = a \cdot (\sigma_1 \parallel b\sigma_2) \cup b \cdot (a\sigma_1 \parallel \sigma_2) \cup [a|b] \cdot (\sigma_1 \parallel \sigma_2)$

with $[a|b] = \{(a|b)\}$ if $a|b \neq \delta$ and $\emptyset$ if $a|b = \delta$.

Thus $\sigma_1 \parallel \sigma_2$ is the set of successful traces obtained by merging and communicating between σ_1 and σ_2. For all traces $\sigma_1 \in F$ and $\sigma_2 \in G$ this set is included in $F \parallel G$ (clause 1). Besides traces $F \parallel G$ contains certain failure pairs $[\sigma, X]$. If either F or G have already terminated, X is just the refusal set of the other, not yet terminated component G or F (clauses 3 and 4). If neither F nor G have terminated, X contains only actions that both F and G can refuse. This suggests $X \subseteq X_1 \cap X_2$ where X_1 and X_2 are the refusal sets of F and G. However, $F \parallel G$ cannot refuse the possible communications between F and G. These communications can only be of the form $(a|b)$ with $a \notin X_1$ and $b \notin X_2$. This explains the condition

$$X \subseteq X_1 \cap X_2 - \{(a|b) \mid a \notin X_1 \wedge b \notin X_2\}$$

for the refusal set X of F || G (clause 2). Note that in case of (a|b) = δ nothing is deducted from $X_1 \cap X_2$.

Clearly, F ⫷ G and F | G are just variations of F || G differing only in their first actions.

(vi) $F \lfloor\!\lfloor G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 \lfloor\!\lfloor \sigma_2\}$
$\cup \{[\varepsilon, X] \mid [\varepsilon, X] \in F\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists [\sigma_1, X_1] \in F, [\sigma_2, X_2] \in G:$
$\sigma \in \sigma_1 \lfloor\!\lfloor \sigma_2$
$\wedge X \subseteq (X_1 \cap X_2) - \{(a|b) \mid a \notin X_1 \wedge b \notin X_2\}\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists \sigma_1 \in F, [\sigma_2, X_2] \in G: \sigma \in \sigma_1 \lfloor\!\lfloor \sigma_2 \wedge X = X_2\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists [\sigma_1, X_1] \in F, \sigma_2 \in G: \sigma \in \sigma_1 \lfloor\!\lfloor \sigma_2 \wedge X = X_1\}$

where $\sigma_1 \lfloor\!\lfloor \sigma_2$ is the set of traces in A* defined inductively by:

$\varepsilon \lfloor\!\lfloor \sigma = \emptyset$,
$a\sigma_1 \lfloor\!\lfloor \sigma_2 = a \cdot (\sigma_1 \| \sigma_2)$.

Until the completion of its first communication F ⫷ G behaves like F. This explains why F ⫷ G inherits all initial failure pairs [ε, X] of F. Afterwards F ⫷ G behaves like F || G.

(vii) $F \mid G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 \mid \sigma_2\}$
$\cup \{[\varepsilon, X] \mid \exists [\varepsilon, X_1] \in F, [\varepsilon, X_2] \in G: X \subseteq A - \{(a|b) \mid a \notin X_1 \wedge b \notin X_2\}\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists [\sigma_1, X_1] \in F, [\sigma_2, X_2] \in G:$
$\sigma \in \sigma_1 \mid \sigma_2$
$\wedge X \subseteq (X_1 \cap X_2) - \{(a|b) \mid a \in X_1 \wedge b \in X_2\}\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists \sigma_1 \in F, [\sigma_2, X_2] \in G: \sigma \in \sigma_1 \mid \sigma_2 \wedge X = X_2\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge \exists [\sigma_1, X_1] \in F, \sigma_2 \in G: \sigma \in \sigma_1 \mid \sigma_2 \wedge X = X_1\}$

where $\sigma_1 \mid \sigma_2$ is the set of traces in A* defined inductively by:

$\varepsilon \mid \sigma_2 = \sigma_1 \mid \varepsilon = \emptyset$,
$a\sigma_1 \mid b\sigma_2 = [a|b] \cdot (\sigma_1 \| \sigma_2)$.

In its first step F | G requires a communication between F and G. Here initially F | G can refuse every set X of actions not containing possible communications between F and G. This explains the condition

$$X \subseteq A - \{(a|b) \mid a \notin X_1 \wedge b \notin X_2\}$$

for the failure pairs $[\varepsilon, X]$. After its first step $F \mid G$ behaves like $F \parallel G$.

(viii) $\partial_H(F) = \{\sigma \mid \sigma \in F$ does not contain any $a \in H\}$
$\cup \{[\sigma, X \cup Y] \mid [\sigma, X] \in F$, s does not contain any $a \in H$, and $Y \subseteq H\}$.

In $\partial_H(F)$ only those traces are successful which do not contain any $a \in H$, and the actions in H can be refused at any moment.

(ix) $a_H(F) = \{a_H(\sigma) \mid \sigma \in F$
$\cup \{[a_H(\sigma), X] \mid a \in X \wedge [\sigma, X \cup H] \in F\}$
$\cup \{[a_H(\sigma), X] \mid a \notin X \wedge [\sigma, X - H] \in F\}$

where the renaming operator a_H is applied pointwise to the elements in σ. A set X can be refused by $a_H(F)$ if $a_H^{-1}(X) = \{b \mid \exists c \in X: a_H(b) = c\}$ can be refused by F.

Except for the different representation of successful termination, the definitions of δ, a, +, $\cdot$, a_H are as for STOP, $a \rightarrow$ SKIP, $\square$, ; and direct image in [BHR84]. The definition of $\parallel$ differs from the parallel composition operators in [BHR84]. In Section 6.2 we will show how to interpret in ACP_r synchronous parallel composition of [BHR84]. The operators $\parallel, \mid, \partial_H$ are not present in [BHR84].

5.3. The failure model.

The *failure model* of ACP_r is now given by the structure $\mathbb{F}(+,\cdot,\parallel,\lfloor\!\lfloor,\mid, \partial_H, a_H, a, \delta)$ $(a \in A)$.

5.3.1. THEOREM. *The failure model of* ACP_r *is isomorphic to the graph model of* ACP_r:

$$\mathbb{H}_\delta(+,\cdot,\parallel,\lfloor\!\lfloor,\mid, \partial_H, a_H, a, \delta)/\equiv_{\mathcal{F}} \;\cong\; \mathbb{F}(+,\cdot,\parallel,\lfloor\!\lfloor,\mid, \partial_H, a_H, a, \delta).$$

PROOF. Consider the mapping $\mathcal{F}: \mathbb{H}_\delta \rightarrow \mathbb{F}$ introduced in Section 2.2. It is clear that $\mathcal{F}$ is well-defined, i.e. that $\mathcal{F}[g] \in \mathbb{F}$ holds for every $g \in \mathbb{H}_\delta$. Also, by Definition 2.2.3, $g \equiv_{\mathcal{F}} h$ iff $\mathcal{F}[g] = \mathcal{F}[h]$ for all $g, h \in \mathbb{H}_\delta$. Thus $\mathcal{F}$ is also well-defined and injective as a mapping

$$\mathcal{F}: \mathbb{H}_\delta/\equiv_{\mathcal{F}} \rightarrow \mathbb{F}$$

(which, by abuse of language, we denote also with $\mathcal{F}$). Now $\mathcal{F}$ is surjective and behaves homomorphically over the operations $+,\cdot,\parallel,\lfloor\!\lfloor,\mid, \partial_H$ and a_H. The proofs of these facts are tedious but follow in a straightforward way from the definitions of these operators on graphs (in 1.2) and the definitions of the corresponding operators on $\mathbb{F}$ (in 5.1). We will not spell out these proofs. Thus $\mathcal{F}$ is the required isomorphism from $\mathbb{H}_\delta(...)$ to $\mathbb{F}(...)$. $\square$

6. ACP_r with 1-1 communication

As a preparation for the subsequent section we now introduce some additional structure on the aphabet A_δ and the communication function $|: A_\delta \times A_\delta \to A_\delta$ of ACP_r.

6.1. 1-1 communication.

First we assume that A (with typical elements $a,b \in A$) is partitioned into $A = C \cup I$ where C (with typical elements $c,d \in C$) is the set of *communicating* actions and I (disjoint from C and with typical elements $i,j \in I$) is the set of *internal* actions. The set I will serve as an auxiliary tool for the communication function $|$.

Secondly, we denote by $\alpha(x)$, the *alphabet* of x, the set of non-δ actions occurring in the closed ACP_r-term x. E.g. $\alpha(a\delta + cd) = \{a,c,d\}$. In subsequent results we will usually be interested in terms x with $\alpha(x) \subseteq C$, i.e. not involving internal, auxiliary actions. Formally, the alphabet of a closed ACP_r-term x is defined by first eliminating the operators $\|, \lfloor\!\lfloor, |, \partial_H$ and a_H from x, using the axioms of ACP_r. (This is possible by virtue of an elimination theorem to this effect proved in [BK84] for ACP; the extra operators a_H in ACP_r present no problem.) The resulting closed term x' contains only the 'basic constructors' + and $\cdot$, and we may further suppose that x' contains no subterm of the form $(p + q)r$ (by some applications of axiom A4 of ACP_r, see Table 1); that is, x' uses only prefix multiplication. Now we define $\alpha(x)$ to be $\alpha(x')$, where $\alpha(x')$ is defined by the following clauses, using induction on the structure of x':

$$\alpha(\delta) = \emptyset$$
$$\alpha(a) = \{a\} \quad (a \in A)$$
$$\alpha(\delta x) = \emptyset$$
$$\alpha(ax) = \{a\} \cup \alpha(x) \quad (a \in A)$$
$$\alpha(x + y) = \alpha(x) \cup \alpha(y).$$

(That $\alpha(x)$ is indeed well-defined in this way, follows from the confluence property of the rewriting procedure used in obtaining x' from x. This fact is for ACP also proved in [BK84] and is easily carried over to ACP_r.)

6.1.1. LEMMA. *For closed terms* x,y *over* ACP_r *with* $\alpha(x), \alpha(y) \subseteq C$ *we have:*

$$\partial_C(x \| y) = \partial_C(x | y).$$

PROOF. It suffices to show that $\partial_C(x \lfloor\!\lfloor y) = \delta$. Recall that x can be normalized in ACP_r to

$$x = \Sigma_i c_i x_i + \Sigma_j d_j$$

with $c_i, d_j \in C$, and with the empty sum Σ denoting δ. Thus

$$x \mathbin{\lfloor\!\lfloor} y = \Sigma_i c_i(x_i \parallel y) = \Sigma_j d_j y$$

which implies $\partial_C(x \mathbin{\lfloor\!\lfloor} y) = \delta$. □

6.1.2. DEFINITION. Assuming the above partition of the alphabet A we say ACP_τ has *1-1 communication* if for the communication merge | there exists a bijection φ: C → C such that $c|\varphi(c) \in I$ for every $c \in C$, and $a|b = \delta$ otherwise.

Note that $c|\varphi(c) \in I$ implies $c|\varphi(c) \neq \delta$. Next, we show that the definitions of parallel composition used in CSP and CCS are typical examples of 1-1 communication.

6.2. Hoare's parallel composition $\|_{\mathcal{H}}$ in CSP.

In [BHR84] Hoare proposes an operation $x \|_{\mathcal{H}} y$ modelling the full synchronization of processes x and y. We shall consider $\|_{\mathcal{H}}$ here within a small subset of the language CSP [BHR84] which we call "CSP". The signature of "CSP" is given by

- the constant STOP,
- unary prefix operators $c \rightarrow$, for $c \in C$,
- the binary infix operators □ and $\|_{\mathcal{H}}$.

Here C is a given set of communication actions, contained in the overall alphabet A.

The semantics of "CSP" is determined by the failures model of [BHR84]. It is based on the failures domain $\mathbb{F}_{BHR}$ consisting of all subsets

$$F \subseteq A^* \times \wp(A)$$

satisfying the closure properties (i)-(iv) discussed in Section 5.1. The additional closure property (v) on traces is not needed here since the failure sets $F \in \mathbb{F}_{BHR}$ contain only failure pairs $[\sigma, X]$.

The failure model assigns to each closed "CSP" term x a failure set $\mathcal{F}_{BHR}[x]$ in the domain $\mathbb{F}_{BHR}$. According to [9] the definition is as follows:

(i) $\mathcal{F}_{BHR}[STOP] = \{[\varepsilon, X] \mid X \subseteq A\}$,

(ii) $\mathcal{F}_{BHR}[c \rightarrow x] = \{[\varepsilon, X] \mid X \subseteq A-\{c\}\} \cup \{[c\cdot\sigma, X] \mid [\sigma, X] \in \mathcal{F}_{BHR}[x]\}$,

(iii) $\mathcal{F}_{BHR}[x \,\square\, y] = \{[\varepsilon, X] \mid [\varepsilon, X] \in \mathcal{F}_{BHR}[x] \cap \mathcal{F}_{BHR}[y]\}$
$\cup \{[\sigma, X] \mid \sigma \neq \varepsilon \wedge [\sigma, X] \in \mathcal{F}_{BHR}[x] \cup \mathcal{F}_{BHR}[y]\}$,

(iv) $\mathcal{F}_{BHR}[x \|_{\mathcal{H}} y] = \{[\sigma, X \cup Y] \mid [\sigma, X] \in \mathcal{F}_{BHR}[x] \wedge [\sigma, Y] \in \mathcal{F}_{BHR}[y]\}$.

The failure model induces the following failure equivalence $\equiv_{\mathcal{F},BHR}$ on closed "CSP" terms x and y:

$$x \equiv_{\mathcal{F},BHR} y \text{ iff } \mathcal{F}_{BHR}[x] = \mathcal{F}_{BHR}[y].$$

We now link these definitions of [BHR84] to our present setting by interpreting "CSP" in ACP_r with 1-1 communication. Let $C = \{c_1,...,c_n\}$. Then we take $A = C \cup I$ with

$$I = \{\underline{c}_1,...,\underline{c}_n\}$$

where the $\underline{c}_i$ ($i = 1,...,n$) are new copies of the actions c_i in C. Furthermore, 1-1 communication is introduced by putting $\varphi(c) = c$ and $c|c = \underline{c}$ for every $c \in C$. The interpretation of "CSP" in ACP_r is given by a mapping $\mathcal{T}$ from closed "CSP" terms into closed ACP_r terms defined as follows:

(i) $\mathcal{T}(STOP) = \delta$,
(ii) $\mathcal{T}(c \rightarrow x) = c \cdot \mathcal{T}(x)$,
(iii) $\mathcal{T}(x \,\square\, y) = \mathcal{T}(x) + \mathcal{T}(y)$,
(iv) $\mathcal{T}(x \|_{\mathcal{H}} y) = C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y)))$

where C_I abbreviates the composite operator $(c_1)_{\{\underline{c}1\}} \circ .... \circ (c_n)_{\{\underline{c}n\}}$, built from the renaming operators $(c_i)_{\{\underline{c}i\}}$ ($i = 1,...,n$) that rename c_i into $\underline{c}_i$.

This interpretation is justified by the following result.

6.2.1. PROPOSITION. *For closed* "CSP" *terms* x

$$\mathcal{F}_{BHR}[x] = \mathcal{F}[\mathcal{T}(x)] \subseteq C^* \times \wp(A)$$

holds where $\mathcal{F}$ *is the* ACP_r *failures model of Section 5. In particular* $\mathcal{F}[\mathcal{T}(x)]$ *does not contain any traces* σ *signalling successful termination, only failure pairs* $[\sigma, X]$.

PROOF. By induction on the structure of x. The cases (i)-(iii) are immediate. Case (iv), parallel composition, is more tedious. It is easy to see that both

$$\mathcal{F}_{BHR}[x \|_{\mathcal{H}} y] \;,\; \mathcal{F}[C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y))] \subseteq C^* \times \wp(A).$$

Hence the closure properties of the failure domains $\mathbb{F}_{BHR}$ and $\mathbb{F}$ respectively, imply

$$[\sigma, X] \in \mathcal{F}_{BHR}[x \|_{\mathcal{H}} y] \text{ iff } [\sigma, X \cup Y] \in \mathcal{F}_{BHR}[x \|_{\mathcal{H}} y],$$

$$[\sigma, X] \in \mathcal{F}[C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y))] \text{ iff } [\sigma, X \cup Y] \in \mathcal{F}[C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y))]$$

for arbitrary $Y \subseteq A - C$. Thus it suffices to show

$$[\sigma, X] \in \mathcal{F}_{BHR}[x \|_{\mathcal{H}} y] \text{ iff } [\sigma, X] \in \mathcal{F}[C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y))]$$

for $\sigma \in C^*$ and $X \subseteq C$.

Let $\underline{\sigma}$ and $\underline{X}$ result from σ and X by replacing pointwise each action c by $\underline{c}$. In particular, we have $\underline{C} = A - C$. Then for $\sigma \in C^*$ and $X \subseteq C$

$$[\sigma, X] \in \mathcal{F}_{BHR}[x \|_{\mathcal{H}} y]$$

iff (induction hypothesis, definition of $\|_{\mathcal{H}}$)

$$\exists X_1 \subseteq C, X_2 \subseteq C:$$
$$[\sigma, X_1] \in \mathcal{F}[\mathcal{T}(x)] \wedge [\sigma, X_2] \in \mathcal{F}[\mathcal{T}(y)] \wedge X \subseteq X_1 \cup X_2$$

iff (definition $\underline{X}$)

$$\exists X_1 \subseteq C, X_2 \subseteq C:$$
$$[\sigma, X_1] \in \mathcal{F}[\mathcal{T}(x)] \wedge [\sigma, X_2] \in \mathcal{F}[\mathcal{T}(y)]$$
$$\wedge \underline{X} \subseteq \{\underline{c} \mid c \in X_1 \cup X_2\}$$

iff (closure properties of the failure domain $\mathbb{F}$)

$$\exists X_1, X_2: \underline{C} \subseteq X_1 \subseteq A \wedge \underline{C} \subseteq X_2 \subseteq A$$
$$\wedge [\sigma, X_1] \in \mathcal{F}[\mathcal{T}(x)] \wedge [\sigma, X_2] \in \mathcal{F}[\mathcal{T}(y)]$$
$$\wedge \underline{X} \subseteq X_1 \cap X_2 - \{\underline{c} \mid c \notin X_1 \cup X_2\}$$

iff (1-1 communication, definition $\|$)

$$[\underline{\sigma}, \underline{X}] \in \mathcal{F}[\mathcal{T}(x) \| \mathcal{T}(y)]$$

iff (definition C_I, ∂_C)

$$[\sigma, X] \in \mathcal{F}[C_I(\partial_C(\mathcal{T}(x) \| \mathcal{T}(y))]$$

This finishes our proof. □

Consequently, for "CSP" the original failure equivalence $\equiv_{\mathcal{F},BHR}$ of [BHR84] coincides with our definition of failure equivalence $\equiv_{\mathcal{F}}$ in Section 2. More precisely:

6.2.2. COROLLARY. *For closed* "CSP" *terms* x *and* y

$$x \equiv_{\mathcal{F},BHR} y \textit{ iff } \mathcal{T}(x) \equiv_{\mathcal{F}} \mathcal{T}(y).$$

For closed "CSP" terms x and y the notions of trace and trace equivalence are defined via the

interpretation in ACP_r:

$$trace(x) = trace(\mathcal{U}(x)),$$
$$x \sim_{tr} y \text{ iff } \mathcal{U}(x) \sim_{tr} \mathcal{U}(y).$$

(Actually, *trace* is in Section 2.1 only defined on graphs; using the operation *graph* from Section 4.1.2 one now defines for a term x, *trace*(x) as *trace*(*graph*(x)).) Using Proposition 6.2.1 the trace set of a term x can also be computed directly from its failure set $\mathcal{F}_{BHR}[x]$:

$$trace(x) = \{\sigma\cdot\delta \mid [\sigma, A] \in \mathcal{F}_{BHR}[x]\}.$$

Recall that in our paper we only consider *complete* traces, either leading to a deadlock δ or to successful termination (not possible for "CSP"). In [BHR84] the word 'trace' is used as well, but it refers to any sequence σ with

$$[\sigma, \emptyset] \in \mathcal{F}_{BHR}[x].$$

Such sequences were called *histories* in Section 2.

6.3. Milner's parallel composition $\|_{\mathcal{M}}$ in CCS.

Since the parallel composition $\|$ in ACP_r can be seen as a generalization of Milner's operation $\|_{\mathcal{M}}$ in CCS [Mi80], it is easy to regain the original definition. As for CSP, we do this within a small subset of CCS which we call "CCS". Milner stipulates that the set C of communicating actions is equipped with a bijection $\bar{\ }: C \to C$ satisfying $\bar{\bar{c}} = c$. Here $\bar{c}$ is called the *matching* action of c. In addition to communicating actions Milner uses a symbol τ denoting the so-called *silent* action. We will write $\underline{\tau}$ because we work here without Milner's τ-laws that make τ silent or invisible (see the discussion below and Section 8). Hence the alphabet for "CCS" will be $A = C \cup \{\underline{\tau}\}$.

The signature of "CCS" consists of

- the constant NIL,
- unary prefix operators $a\cdot$, for $a \in A$,
- unary postfix operators $\backslash H$, for $H \subseteq C$,
- the binary infix operator + and $\|_{\mathcal{M}}$.

Informally, $x \|_{\mathcal{M}} y$ denotes the nondeterministic interleaving of x and y, plus the communication of x and y via matching actions which then yield $\underline{\tau}$ as a result. Following [Mi80], this can be expressed by the infinite axiom scheme

(*) $$(\Sigma_i a_i x_i) \|_{\mathcal{M}} (\Sigma_j b_j y_j) =$$
$$\Sigma_i a_i(x_i \|_{\mathcal{M}} y) + \Sigma_j b_j(x \|_{\mathcal{M}} y_j) + \Sigma_{a_i = \bar{b}_j} \underline{\tau}\cdot(x_i \|_{\mathcal{M}} y_j)$$

where $x = \sum_i a_i x_i$ and $y = \sum_j b_j y_j$.

We shall define the semantics of $\|_{\mathcal{M}}$ via an interpretation $\mathcal{I}$ of "CCS" in ACP_r with 1-1 communication. To this end, take $I = \{\underline{\tau}\}$ and define

$$\varphi(c) = \overline{c} \text{ and } c\overline{c} = \underline{\tau}.$$

Then $\mathcal{I}$ is rather trivial:

(i) $\mathcal{I}(NIL) = \delta$
(ii) $\mathcal{I}(a \cdot x) = a \cdot \mathcal{I}(x)$
(iii) $\mathcal{I}(x \backslash H) = \partial_H(\mathcal{I}(x))$
(iv) $\mathcal{I}(x + y) = \mathcal{I}(x) + \mathcal{I}(y)$
(v) $\mathcal{I}(x \|_{\mathcal{M}} y) = \mathcal{I}(x) \| \mathcal{I}(y)$.

Note that the auxiliary operations $\|_$ and | in ACP_r serve to replace the infinite axiom scheme (*) by finitely many ACP_r axioms.

In [Mi80] Milner studies CCS terms under the (weak) bisimulation equivalence [Pa83], but here we shall study "CCS" under the failure equivalence. For closed "CCS" terms x and y we define the notions of failure equivalence, trace equivalence and alphabet via the interpretation $\mathcal{I}$ in ACP_r:

$x \equiv_F y$ iff $\mathcal{I}(x) \equiv_F \mathcal{I}(y)$,
$x \sim_{tr} y$ iff $\mathcal{I}(x) \sim_{tr} \mathcal{I}(y)$,
$\alpha(x) = \alpha(\mathcal{I}(x))$.

In general, these definitions are not quite appropriate for CCS because τ should be silent or invisible; more formally τ should be subject to Milner's τ-laws. In the above interpretation of "CCS" $\underline{\tau}$ remains visible, i.e. recorded in the traces and failure pairs. The reason for this clash is that CCS indivisibly couples parallel composition $\equiv_F$ and τ whereas we decided to separate failure equivalence from τ.

However, we can regain the spirit of CCS if we restrict the failure equivalence to $\underline{\tau}$-*free* "CCS" terms x and y, i.e. with

$$\underline{\tau} \notin \alpha(x), \alpha(y).$$

Unfortunately, $\underline{\tau}$-free "CCS" terms are not closed under parallel composition $\|_{\mathcal{M}}$. Therefore we shall consider also a modified trace set

$$trace_{\underline{\tau}}(x)$$

for "CCS" terms x which results from *trace*(x) by deleting in every trace $\sigma\cdot\delta \in$ *trace*(x) all occurrences of τ in σ. Then $trace_\tau$(x) represents the set of complete traces in the sense of CCS. For example,

$$trace(cNIL \parallel_M \bar{c}NIL) = \{c\bar{c}\delta, \bar{c}c\delta, \tau\delta\},$$
$$trace_\tau(cNIL \parallel_M \bar{c}NIL) = \{c\bar{c}\delta, \bar{c}c\delta, \delta\}.$$

7. The maximal trace respecting congruence

In Section 4 (Proposition 4.2.3) it was shown that failure equivalence $\equiv_F$ is a congruence with respect to the operators of ACP_r. In this section we will prove that for ACP_r with 1-1 communication failure equivalence is in fact the maximal trace respecting congruence. This implies a full abstraction result for the failure model of Section 5. But first let us introduce the relevant concepts.

7.1. Preliminaries.

Let Σ be a signature with Ter(Σ) denoting the set of closed terms over Σ. By Ter(Σ)[ξ] we denote the set of terms over Σ with ξ as free variable. These terms are called *contexts* and are typically written as $C[\xi]$.

Let $\mathcal{T} \subseteq$ Ter(Σ). A *congruence for* $\mathcal{T}$ is an equivalence relation $\equiv$ on $\mathcal{T}$, such that

$$x \equiv y \text{ implies } C[x] \equiv C[y]$$

for all terms $x,y \in \mathcal{T}$ and contexts $C[\xi] \in$ Ter(Σ)[ξ] with $C[x], C[y] \in \mathcal{T}$. A congruence $\equiv$ for $\mathcal{T}$ is *trace respecting* if

$$x \equiv y \text{ implies } trace(x) = trace(y)$$

for all $x,y \in \mathcal{T}$. A trace respecting congruence $\equiv$ for $\mathcal{T}$ is called *maximal* if for all $x,y \in \mathcal{T}$, $x \not\equiv y$ implies that there exists some context $C[\xi] \in$ Ter(Σ)[ξ] with $C[x], C[y] \in \mathcal{T}$ and $trace(C[x]) \neq trace(C[y])$.

7.1.1. PROPOSITION. *For each* $\mathcal{T} \subseteq$ Ter(Σ) *the maximal trace respecting congruence for* $\mathcal{T}$ *exists and is unique.*

PROOF. *Uniqueness:* Suppose $\equiv_1$ and $\equiv_2$ are different maximal trace respecting congruences on $\mathcal{T}$. Then for some $x,y \in \mathcal{T}$ we have

$$x \equiv_1 y, \text{ but } x \not\equiv_2 y.$$

Since $\equiv_1$ is a trace respecting congruence on $\mathcal{T}$, *trace*($\mathbb{C}$[x]) = *trace*($\mathbb{C}$[y]) holds for every context $\mathbb{C}[\xi] \in Ter(\Sigma)[\xi]$ with $\mathbb{C}$[x], $\mathbb{C}$[y] $\in \mathcal{T}$. But this contradicts the maximality of $\not\equiv_2$.

Existence: Define $\equiv$, a binary relation on $\mathcal{T}$, as follows: $x \equiv y$ iff for every context $\mathbb{C}[\xi] \in Ter(\Sigma)[\xi]$ with $\mathbb{C}$[x], $\mathbb{C}$[y] $\in \mathcal{T}$, *trace*($\mathbb{C}$[x]) = *trace*($\mathbb{C}$[y]) holds.

It is easy to see that $\equiv$ is a trace respecting congruence for $\mathcal{T}$; maximality follows from its definition. □

7.2. A characterisation of failure equivalence.

Let us now turn to ACP_r. We write $Ter(ACP_r)$ instead of $Ter(\Sigma)$. From Section 4 we know that failure equivalence $\equiv_{\mathcal{F}}$ is a trace respecting congruence for $Ter(ACP_r)$. (For the sake of convenience, we have identified here the semantical notion $\equiv_{\mathcal{F}}$ with the equivalence induced by $\equiv_{\mathcal{F}}$ on $Ter(ACP_r)$ via the correspondence between process graphs and terms, explained in Section 4.1.) Thus for ACP_r in general we have

$$\equiv_{\mathcal{F}} \subseteq \equiv_{max}$$

with $\equiv_{max}$ denoting the maximal trace respecting congruence for $Ter(ACP_r)$. If we specialize ACP_r to the case of 1-1 communication, we can actually prove

$$\equiv_{\mathcal{F}} = \equiv_{max}$$

and thus arrive at a very pleasing characterization of failure equivalence:

7.2.1. THEOREM. *Consider* ACP_r *with 1-1 communication. Then failure equivalence* $\equiv_{\mathcal{F}}$ *is the maximal trace respecting congruence for the set* $\mathcal{T}_C$ *of all closed terms* x *over* ACP_r *with alphabet* $\alpha(x) \subseteq C$.

PROOF. Suppose $x \not\equiv_{\mathcal{F}} y$, i.e. $\mathcal{F}[x] \neq \mathcal{F}[y]$ holds for $x,y \in \mathcal{T}_C$. If *trace*(x) ≠ *trace*(y), the trivial context $\mathbb{C}[\xi] = \xi$ will do. Now suppose that *trace*(x) = *trace*(y) holds. Because of $x \not\equiv_{\mathcal{F}} y$ we can assume without loss of generality that there exists a failure pair [σ, X] with

$$[\sigma, X] \in \mathcal{F}[x], [\sigma, X] \notin \mathcal{F}[y].$$

By the definition of $\mathcal{F}$, $[\sigma, X] \in \mathcal{F}[x]$ implies that there exists some ready pair $(\sigma, Z) \in \mathcal{R}[x]$ with $X \subseteq Z$. Note that $Z \neq \emptyset$. Suppose we had $(\sigma, \emptyset) \in \mathcal{R}[x]$. Then $\sigma\delta \in$ *trace*(x) = *trace*(y) and $(\sigma,\emptyset) \in \mathcal{R}[y]$.Thus $[\sigma, C] \in \mathcal{F}[y]$ and therefore also $[\sigma, X] \in \mathcal{F}[x]$. Contradiction.

Trace equivalence of x and y implies that there exists a ready pair $(\sigma, Y) \in \mathcal{R}[y]$ with $Y \neq \emptyset$. Again by the definition of $\mathcal{F}$, $[\sigma, X] \notin \mathcal{F}[y]$ implies that for every such ready pair $(\sigma, Y) \in \mathcal{R}[y]$ there exists some $d \in X \cap Y$. Now consider a context of the form

$$\mathbb{C}[\xi] = (c_{1\{i1\}} \circ \ldots \circ c_{n\{in\}} \circ \partial_C)(x \parallel \varphi(\sigma) \cdot \Sigma\, \varphi(d) \cdot \delta)$$

where the sum Σ is taken over all $d \in X \cap Y$ such that $(\sigma, Y) \in \mathcal{R}[y]$. Furthermore $I = \{i_1,\ldots,i_n\}$, $c_1,\ldots,c_n \in C$, φ is the bijection describing the 1-1 communication in ACP_r and $\varphi(\sigma)$ is the result of applying φ pointwise to σ. Note that $\mathbb{C}[\xi]$ is uniquely determined by x and y except for the choice of the $c_1,\ldots,c_n$ in the renaming operators. Note that indeed $\mathbb{C}[x], \mathbb{C}[y] \in \mathcal{T}_C$ due to the presence of operators ∂_C and $c_{j\{ij\}}$ in $\mathbb{C}[\xi]$. We now claim that

$$(c_{1\{i1\}} \circ \ldots \circ c_{n\{in\}})(\sigma|\varphi(\sigma)) \cdot \delta \in \mathit{trace}(\mathbb{C}[x]), \notin \mathit{trace}(\mathbb{C}[y])$$

where $\sigma|\varphi(\sigma)$ is understood by applying $|$ pointwise to σ and $\varphi(\sigma)$.

To prove this claim we first state a general observation about ready sets $\mathcal{R}[z]$ of closed terms z over ACP_r. Let $\sigma = a_1 \ldots a_m$ and $Z = \{b_1,\ldots,b_n\}$. Then $(\sigma, Z) \in \mathcal{R}[z]$ iff there exist $x_1,\ldots,x_m,y_1,\ldots,y_n \in \mathrm{Ter}(ACP_r)$ with

$$ACP_r \vdash x = a_1(a_2 \ldots (a_m(b_1y_1 + \ldots + b_ny_n) + x_m) \ldots + x_2) + x_1.$$

This observation is obvious from Sections 3 and 4.

Next we recall from Lemma 6.1.1 that due to the encapsulation ∂_C we can replace the general parallel composition $\parallel$ in $\mathbb{C}[\xi]$ by the communication operator $|$ which enforces synchronization.

Combining these two facts, it is easy to calculate that $(\sigma, Z) \in \mathcal{R}[x]$ with $X \subseteq Z$ yields

$$(c_{1\{i1\}} \circ \ldots \circ c_{n\{in\}})(\sigma|\varphi(\sigma)) \cdot \delta \in \mathit{trace}(\mathbb{C}[x]).$$

Now suppose that this trace is also present in $\mathit{trace}(\mathbb{C}[y])$. Since ACP_r allows only 1-1 communication, there exists a history $\sigma \in C^*$ such that every ready pair $(\sigma, Y) \in \mathcal{R}[y]$ satisfies $X \cap Y = \emptyset$. Contradiction. This finishes our proof. □

7.3. Application to CSP and CCS.

The characterization of failure equivalence for ACP_r yields corresponding results for the subsets "CSP" and "CCS" of [BHR84] and [Mi80].

7.3.1. COROLLARY. *For closed* "CSP" *terms the failure equivalence* $\equiv_{\mathcal{F},BHR}$ *of* [9] *is the maximal trace respecting congruence.*

PROOF. Via the interpretation $\mathcal{T}$ the failure equivalence $\equiv_{\mathcal{F},BHR}$ is a trace respecting congruence for "CSP". To show maximality, suppose $x \not\equiv_{\mathcal{F},BHR} y$ for closed terms x and y. Then $\mathcal{T}(x) \not\equiv_{\mathcal{F}} \mathcal{T}(y)$ by Corollary 6.2.2. Since $\alpha(\mathcal{T}(x)), \alpha(\mathcal{T}(y)) \subseteq C$, Theorem 7.2.1 applies and yields a context $\mathbb{C}[\xi]$ in ACP_r with

$$\mathbb{C}[\mathbb{T}(x)] \not\leftrightarrow_{tr} \mathbb{C}[\mathbb{T}(y)].$$

Looking at the proof of Theorem 7.2.1 we see that $\mathbb{C}[\xi]$ can be expressed in "CSP", i.e. there exists a context $\mathbb{C}'[\xi]$ in "CSP" with

$$I(\mathbb{C}')[\xi] = \mathbb{C}[\xi]$$

where we stipulate $\mathbb{T}(\xi) = \xi$. Thus

$$\mathbb{T}(\mathbb{C}')[\mathbb{T}(x)] \not\leftrightarrow_{tr} \mathbb{T}(\mathbb{C}')[\mathbb{T}(y)].$$

Since $\mathbb{T}$ is defined by structural induction, we have $\mathbb{T}(\mathbb{C}')[\mathbb{T}(x)] = \mathbb{T}(\mathbb{C}'[x])$ and likewise for y. Thus

$$\mathbb{C}'[x] \not\leftrightarrow_{tr} \mathbb{C}'[y]$$

by the definition of trace equivalence for "CSP". □

Due to the differences of τ and $\underline{\tau}$ in CCS and ACP_r (see Section 6.3), we can characterize failure equivalence only for $\underline{\tau}$-free "CCS" terms.

7.3.2. COROLLARY. *On the subset of closed, $\underline{\tau}$-free* "CCS" *terms failure equivalence $\leftrightarrow_{\mathcal{F}}$ coincides with the maximal trace respecting congruence defined for full* "CCS". *This result holds for both notions of trace introduced for* "CCS" *terms, viz. trace(.) and $trace_{\underline{\tau}}$(.).*

PROOF. Via the interpretation $\mathbb{T}$ failure equivalence $\leftrightarrow_{\mathcal{F}}$ is a trace respecting congruence for "CCS". This holds for the original definition of *trace*(.), but since

$$trace(x) = trace(y) \text{ implies } trace_{\underline{\tau}}(x) = trace_{\underline{\tau}}(y),$$

it holds for $trace_{\underline{\tau}}$(.) as well.

Now consider two closed, $\underline{\tau}$-free "CCS" terms x,y such that $x \not\leftrightarrow_{\mathcal{F}} y$, i.e. $\mathbb{T}(x) \not\leftrightarrow_{\mathcal{F}} \mathbb{T}(y)$. Since $\underline{\tau}$-freeness means $\alpha(\mathbb{T}(x)), \alpha(\mathbb{T}(y)) \subseteq C$, the proof technique for Theorem 7.2.1 applies and yields an ACP_r context of the form

$$\mathbb{C}[\xi] = \partial_C(\xi \parallel \mathbb{T}(z))$$

where z is a closed, $\underline{\tau}$-free "CCS" term such that for some $n \geq 0$

$$\underline{\tau}^n \cdot \delta \in trace(\mathbb{C}[\mathbb{T}(x)]), \notin trace(\mathbb{C}[\mathbb{T}(y)]).$$

Note that in the definition of $\mathbb{C}[\xi]$ we deviate slightly from Theorem 7.2.1 and omit the renaming

operator which would yield here $c_{\{\underline{\tau}\}}$ for some $c \in C$. The reason is that τ (respectively $\underline{\tau}$) cannot be renamed in Milner's [Mi80] (and hence "CCS").

The above $\mathbb{C}[\xi]$ can be translated back into the "CCS" context

$$\mathbb{C}'[\xi] = (\xi \|_{\mathcal{M}} z)\backslash C,$$

yielding

$$\underline{\tau}^n \cdot \delta \in \mathit{trace}(\mathbb{C}'[x]), \notin \mathit{trace}(\mathbb{C}'[y])$$

and thus

$$\delta \in \mathit{trace}_{\underline{\tau}}(\mathbb{C}'[x]), \notin \mathit{trace}_{\underline{\tau}}(\mathbb{C}'[y]).$$

This proves the maximality of failure equivalence for $\underline{\tau}$-free "CCS" terms with respect to both notions of trace. □

Thus the (proof of) Theorem 7.2.1 gives a uniform argument for the communication mechanisms of both "CSP" and "CCS".

7.3.3. REMARK. (Comparison with the work of De Nicola & Hennessy [DH84].)
We have proved that (under a restricted communication format) processes are failure equivalent if and only if they cannot be separated by any context where 'separated' refers to the criterion of having different traces. This characterisation is easy to understand as it involves only the notions of *trace* and *context*. It is interesting to compare our result with a result in [DH84]. Since the settings are quite different (here finite processes in ACP_r, there CCS with recursion, τ-steps and an additional constant Ω denoting the undefined state), we state the comparison for the greatest common denominator of ACP_r and CCS, viz. the language "CCS" of Section 6.3.

De Nicola and Hennessy [DH84] set up a notion of *testing* and consider two processes p and q as equivalent if and only if they pass exactly the same tests. This idea of testing is very appealing, but the formal definitions are somewhat more technical. Both processes and tests are just terms over the signature of "CCS". However, in the alphabet A one assumes a distinguished action ω which may appear in tests only. The action ω is interpreted as reporting success; it is needed in the definition of a process passing a test. Due to the restriction to "CCS", we can phrase De Nicola & Hennessy's definition as follows.

For "CCS" terms p,q,r and actions $a \in A$ we write:

$$p \rightarrow_a q \text{ if } \exists r\text{: ``CCS''} \vdash p = a \cdot q + r$$
$$p \rightarrow_a \text{ if } \exists q\text{: } p \rightarrow_a q.$$

Intuitively, $p \rightarrow_a q$ states that p can perform an action a and then behave like q. A *computation* is a sequence of "CCS" terms of the form

$$p_1 \rightarrow_{\underline{\tau}} p_2 \rightarrow_{\underline{\tau}} \cdots \rightarrow_{\underline{\tau}} p_n;$$

it is called *maximal* if there is no "CCS" term q with $p_n \rightarrow_{\underline{\tau}} q$. Since "CCS" does not include recursion, any computation is finite here.

There are two forms of a process p passing a test t:

(i) p *may pass* t if there exists a computation

$$p \|_{\mathcal{M}} t = p_1 \|_{\mathcal{M}} t_1 \rightarrow_{\underline{\tau}} \cdots \rightarrow_{\underline{\tau}} p_n \|_{\mathcal{M}} t_n$$

with $t_n \rightarrow_{\omega}$, or equivalently if there exists some $n \geq 0$ with

$$\underline{\tau}^n \cdot \omega \in \text{trace}(p \|_{\mathcal{M}} t),$$

(ii) p *must pass* t if whenever

$$p \|_{\mathcal{M}} t = p_1 \|_{\mathcal{M}} t_1 \rightarrow_{\underline{\tau}} \cdots \rightarrow_{\underline{\tau}} p_n \|_{\mathcal{M}} t_n$$

is a *maximal computation* then there exists some m with $1 \leq m \leq n$ and $t_m \rightarrow_{\omega}$.

Thus a term t_n that can perform an ω-action serves as a criterion for success. For examples of (i) and (ii) we refer to [DH84].

Then De Nicola and Hennessy [DH84] introduce three so-called *testing equivalences* on processes p,q:

(i) $p \simeq_1 q$ if for every test t: p may pass t iff q may pass t.
(ii) $p \simeq_2 q$ if for every test t: p must pass t iff q must pass t.
(iii) $p \simeq_3 q$ if $p \simeq_1 q$ and $p \simeq_2 q$.

It is now very interesting that for $\underline{\tau}$-free "CCS" the strong testing equivalence coincides with the failure equivalence $\equiv_{\mathcal{F}}$. This is an immediate consequence of Corollary 6.2.6 of [DH84] stated for the class of so-called strongly divergent CCS terms which in particular includes all $\underline{\tau}$-free "CCS" terms. Thus at least for $\underline{\tau}$-free "CCS" terms we have a pleasing convergence of ideas:

strong testing equivalence = failure equivalence = maximal trace respecting congruence.

Conceptually, we find the notion of a maximal trace respecting congruence simpler than the definition of passing a test.

7.4. Full abstraction.

The notion of *full abstraction* is due to Milner [Mi77] (see also [HP79, Pl77). It is a relationship between models (of an axiomatic system) and equivalence relations (on the terms of that system) whose definition is motivated by the following question:

Under what circumstances can we replace a term x *by a term* y *without noticing this change by a given equivalence* ≡*?*

Using the notion of a context introduced above, this question amounts to:

Under what conditions on x *and* y *do we have* $\mathbb{C}[x] \equiv \mathbb{C}[y]$ *for every context* $\mathbb{C}[\xi]$*?*

Full abstraction can be seen as looking for a sufficient and necessary condition that answers this question. Formally, we state:

7.4.1. DEFINITION. A model $\mathcal{M}$ for $\mathcal{T} \subseteq \mathrm{Ter}(\Sigma)$ is called *fully abstract with respect to an equivalence relation* ≡ on $\mathcal{T}$ if for all terms $x,y \in \mathcal{T}$:

$\mathcal{M}[x] = \mathcal{M}[y]$ iff $\mathbb{C}[x] \equiv \mathbb{C}[y]$ holds for every context $\mathbb{C}[\xi] \in \mathrm{Ter}(\Sigma)[\xi]$ with $\mathbb{C}[x], \mathbb{C}[y] \in \mathcal{T}$.

Thus a fully abstract model $\mathcal{M}$ optimally fits the equivalence ≡ in the sense that it just makes the identifications on terms that are forced by ≡. Usually, it is quite difficult to discover fully abstract models (see [HP79, Mi77, Pl77]), but for the failure model $\mathcal{F} = \mathbb{F}(+,\cdot,\|,\lfloor\!\lfloor,\mid,\partial_H,a_H,a,\delta)$ $(a \in A)$ of Section 5 and the trace equivalence $\sim_{tr}$ of Section 2 we can now state such a result.

7.4.2. THEOREM. *Consider* ACP_r *with 1-1 communication. Then for the set* $\mathcal{T}_c$ *of all closed terms* x *over* ACP_r *with alphabet* $\alpha(x) \subseteq C$ *the failure model* $\mathcal{F}$ *is fully abstract with respect to the trace equivalence* $\sim_{tr}$.

PROOF. By Definition 7.3.1, it suffices to show that for all $x,y \in \mathcal{T}_c$:

$$\mathcal{F}[x] = \mathcal{F}[y] \text{ iff } x \equiv_{max} y$$

where $\equiv_{max}$ is the maximal trace respecting congruence. But this is immediate from Theorem 7.2.1. □

7.4.3. COROLLARY. *For the set of closed* "CSP" *terms the failure model* $\mathcal{F}_{BHR}$ *of* [9] *is fully abstract with respect to the trace equivalence* $\sim_{tr}$.

For "CCS" we cannot state the analogous result due to the τ mismatch discussed above.

8. Processes with recursion and abstraction: bisimulation versus failure equivalence

8.1. Preliminaries.

In the preceeding sections we have been exclusively concerned with the failure semantics for finite processes withhout abstraction, i.e. not involving τ-steps. In this section we will set aside that restriction and comment also on infinite (recursive) processes with abstraction, as regards bisimulation and failure equivalence. The crucial point is the way in which infinite sequences of

τ-steps in a process are treated.

In the failure semantics proposed in [BHR84], all processes having an infinite τ-sequence from the root are set equal (to the process CHAOS). The notion of bisimulation is more discriminating. The advantage is that models obtained by bisimulation equivalence satisfy a useful abstraction principe: *Koomen's fair abstraction rule* (KFAR) as introduced in [BK84b]. Roughly, this rule gives a way of simplifying processes by elimination of (some) infinite τ-sequences. This elimination can be understood as *fairness* of (visible) actions over silent τ-steps. A more precise description is given below. (Of course, setting all processes having an infinite τ-sequence from the root equal to CHAOS also eliminates infinite τ-sequences, but then all information is lost.)

Since KFAR is a very useful tool for system verification (e.g. in [BK84b] it was used to verify an alternating bit protocol), it is natural to ask wether KFAR is also compatible with the somewhat simpler failure semantics. More precisely, one can ask whether there exist a process model which for finite processes agrees with the failure semantics and for infinite processes satisfies KFAR. Interestingly, it turns out that such a model does not exist. To prove this result, we will formulate a set of assumptions embodying failure semantics and KFAR, and derive an inconsistency. Formally, the inconsistency arises from the following extension of the axiom system considered above:

ACP_τ + R1,2 + S +
Milner's τ-laws + axioms for abstraction operators +
KFAR+
RSP (recursive specification principle).

Here RSP is the assumption that guarded systems of recursion equations have a solution, which is moreover unique.

Now by virtue of our axiomatic approach we can pinpoint the origin of the inconsistency derived below with some accuracy. It turns out that the failure of KFAR in failure semantics holds already in ready semantics, and moreover that communication does not play a role in the inconsistency. That is, the inconsistency already appears in the subsystem

BPA + T1 + TI1-5 + R1 + KFAR + RSP

which we will explain now. BPA, for *basic process algebra*, consists of the axioms A1-5 of ACP_τ, which specify the properties of + and $\cdot$. T1 is the simplest of Milner's τ-laws [Mi80] (see Table 5 below).

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$x\tau = x$	T1
$\tau_I(\tau) = \tau$	TI1
$\tau_I(a) = a$ if $a \notin I$	TI2
$\tau_I(a) = \tau$ if $a \in I$	TI3
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	TI4
$\tau_I(xy) = \tau_I(x) \cdot \tau_I(y)$	TI5

BPA + T1 + TI1-5

Table 6

In addition, Table 6 contains axioms TI1-5; these specify the abstraction operators τ_I where $I \subseteq A$ is a set of *internal* actions as simple renaming operators (cf.[BK84c] and [BK86a,b]).

R1 is the axiom for the readiness semantics (see Tables 3,4):

$$a(bx + u) + a(by + v) = a(bx + by + u) + a(bx + by + v).$$

The *recursive specification principle* RSP states that guarded systems E of recursive equations have unique solutions (see [BK84b] or [BBK85]):

$$\frac{E(x_1,\ldots,x_n),\ E(y_1,\ldots,y_n),\ E \text{ guarded}}{x_1 = y_1}$$

Informally, 'guarded' means that every recursive occurrence of x_i in E is preceded by an action different from τ. For example, the system

$$\begin{cases} x_1 = ax_2 + bx_2 \\ x_2 = c(x_1 + x_2) + d \end{cases}$$

is guarded and thus has a unique solution.

We will now explain KFAR. For each $n \geq 1$, we have a version $KFAR_n$. $KFAR_1$ is as follows:

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

The premise of $KFAR_1$ says that x has an infinite i-trace; see Figure 19. Now $KFAR_1$ expresses the fact that x makes *fair* choices along its infinite i-trace, i.e. performing x entails at most finitely

many choices against y.

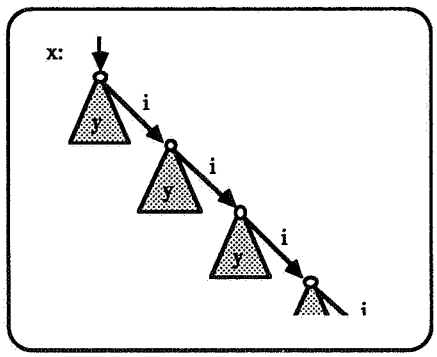


Figure 19

We may note here the necessity of the abstraction operator τ_I in $KFAR_1$: From $x = \tau x + y$ it does *not* follow that $x = \tau \cdot \tau_I(y)$, since the equation $x = \tau x + y$ has infinitely many solutions (see [BK84c] or [BK86a]).

The version of KFAR for n = 2 is

$$\frac{x_1 = ix_2 + y_1,\ \ x_1 = ix_2 + y_1\ (i,j \in I)}{\tau_I(x_1) = \tau \cdot \tau_I(y_1 + y_2)}$$

In the general formulation of $KFAR_n$ the premise displays an "I-cycle" of length n. For a precise formulation we refer to [BK84b] or [BBK85].

Note that except for KFAR all assumptions in BPA_τ + TI1-5 + R1 + RSP are valid for failure semantics. To see that the τ-laws TI-3 (of which only the first one is needed for the derivation of the contradiction below) are valid for failure semantics, we refer to [Br83] who gives axioms describing failure semantics for finite processes involving τ-steps; these axioms imply the τ-laws.

8.2. The inconsistency of failure semantics with KFAR.

We will now derive the announced contradiction. It is important to notice that this contradiction is entirely insensitive to how failure semantics works with processes that contain τ-steps.

Consider the following systems of guarded recursion equations:

$$E_1 \quad \begin{cases} x = ax_1 + ax_2 \\ x_1 = c + bx_2 \\ x_2 = d + bx_1 \end{cases}$$

and

$$E_1 \quad \begin{cases} y = ay_1 + ay_2 \\ y_1 = c + by_2 \\ y_2 = d + by_1 \end{cases}$$

The systems E_1, E_2 have solutions x,y which can be depicted as in Figure 20:

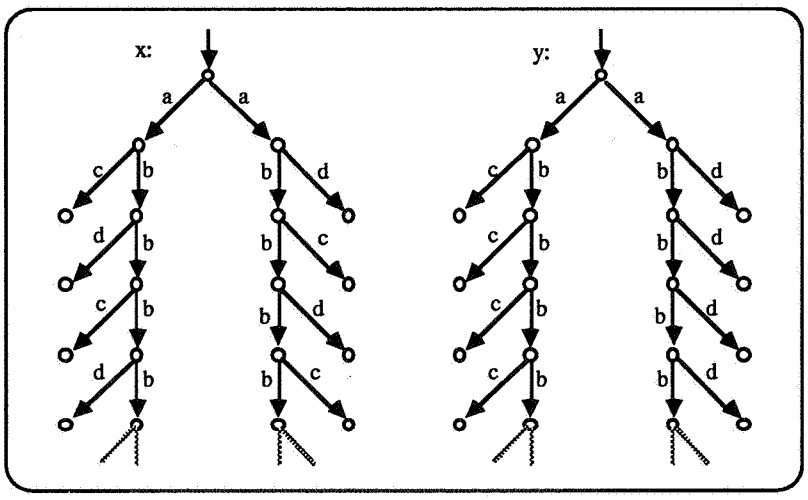


Figure 20

Claim: x and y are failure equivalent.

Intuitively this may be clear since (as demonstrated in Section 3.1) axiom R1 amounts to placing 'crosses'; from the graphs for x,y above we can thus obtain equivalent graphs as in Figure 21. These two graphs are in fact identical.

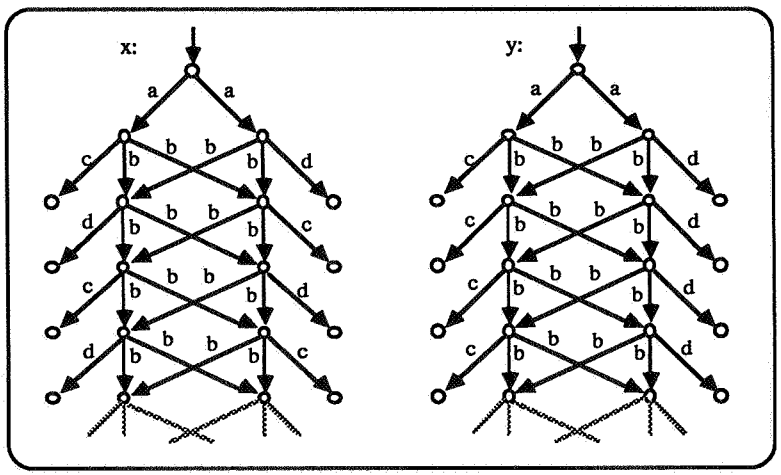


Figure 21

Formally: *Proof of the claim.* Consider the system E_3 of guarded recursion equations:

$$E_3 \quad \begin{cases} z = az_1 + az_{21} \\ z_1 = c + bz_1 + bz_2 \\ z_2 = d + bz_1 + bz_2 \end{cases}$$

(This system corresponds with the graph in Figure 21.) Now

$$x = ax_1 + ax_2 = a(c + bx_2) + a(d + bx_1) = \text{ (by R1)}$$
$$a(c + bx_2 + bx_1) + a(d + bx_1 + bx_2) = az_1' + az_2'$$

where

$$z_1' = c + bx_2 + bx_1 \text{ and } z_2' = d + bx_1 + bx_2.$$

Further,

$$z_1' = c + bx_2 + bx_1 = c + b(d + bx_1) + b(c + bx_2) = \text{ (by R1)}$$
$$c + b(c + bx_2 + bx_1) + b(d + bx_1 + bx_2) =$$
$$c + bz_1' + bz_2'$$

and likewise

$$z_2' = d + bz_1' + bz_2'.$$

So (x, z_1', z_2') satisfies E_3. A similar computation shows that (y, z_1'', z_2'') where

$$z_1'' = c + by_1 + by_2$$
$$z_2'' = d + by_1 + by_2$$

satisfies E_3. Hence by RSP,

$$(x, z_1', z_2') = (y, z_1'', z_2'') = (z, z_1, z_2),$$

in particular $x = y$. This proves the claim.

In order to derive the inconsistency we will abstract from b, by means of $\tau_{\{b\}}$, in x and y. This yields corresponding process graphs as in Figure 22.

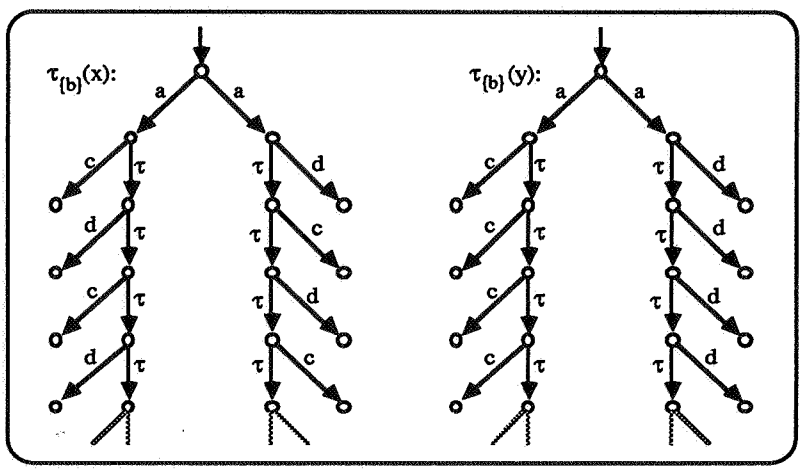


Figure 22

Next we apply KFAR on $\tau_{\{b\}}(x)$ and $\tau_{\{b\}}(y)$ and obtain $a(c + d)$ and $ac + ad$, respectively. This can

be seen graphically: KFAR shrinks the infinite τ-traces to a point, obtaining the graphs as in Figure 23.

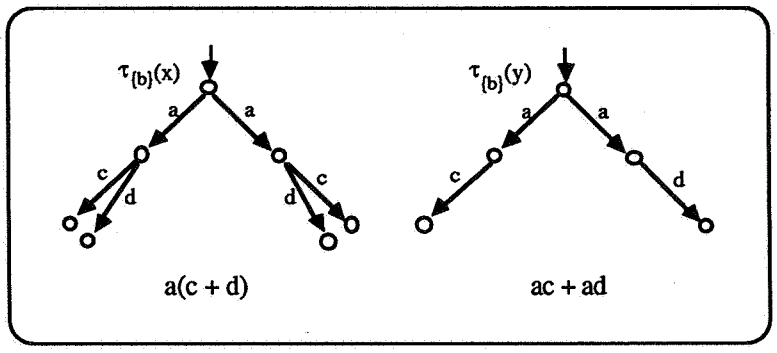


Figure 23

Formally:

$$\tau_{\{b\}}(x) = \tau_{\{b\}}(ax_1 + ax_2) = a\cdot\tau_{\{b\}}(x_1) + a\cdot\tau_{\{b\}}(x_2) \qquad (*)$$

Further,

$$x_1 = bx_2 + c,\ x_2 = bx_1 + d$$

yields by $KFAR_2$:

$$\tau_{\{b\}}(x_1) = \tau\cdot\tau_{\{b\}}(c + d) = \tau(c + d)$$
$$\tau_{\{b\}}(x_2) = \tau\cdot\tau_{\{b\}}(c + d) = \tau(c + d)$$

Hence from (*):

$$\tau_{\{b\}}(x) = a\tau(c + d) + a\tau(c + d) = \text{ (by T1 in Table 6)}$$
$$a(c + d) + a(c + d) = a(c + d).$$

Next consider y:

$$\tau_{\{b\}}(y) = a\cdot\tau_{\{b\}}(y_1) + a\cdot\tau_{\{b\}}(y_2) \qquad (**)$$

Now $y_1 = by_1 + c$ yields by $KFAR_1$: $\tau_{\{b\}}(y_1) = \tau c$; similarly $\tau_{\{b\}}(y_2) = \tau d$. Hence from (**):

$$\tau_{\{b\}}(y) = a\tau c + a\tau d = ac + ad.$$

So, since x = y, we have proved a(c + d) = ac + ad. But a(c + d) and ac + ad are not failure equivalent.

8.3. Further results.

The above inconsistency proves that the advantages of Koomen's fair abstraction rule KFAR cannot be combined with the simplicity of failure semantics. We have investigated this dichotomy

further and were pleased to find a weaker fair abstraction rule called KFAR$^-$ which *is consistent* with (finite) failure semantics, and which is still useful for many process verifications. More precisely, the new rule is consistent with a version of Brookes, Hoare and Roscoe's failure semantics [BHR84] without catastrophic divergence, i.e. that does not identify processes having an infinite τ-sequence from the root with the process CHAOS. The details and applications of the new rule KFAR$^-$ can be found in [BKO86].

References

[BBK85] BAETEN, J.C.M., J.A. BERGSTRA & J.W. KLOP, *On the consistency of Koomen's Fair Abstraction Rule*, Report CS-R8511, Centre for Mathematics and Computer Science, Amsterdam 1985. To appear in TCS 51 (1/2).

[BK83] BERGSTRA, J.A. & J.W. KLOP, *An abstraction mechanism for process algebras*, Report IW 231/83, Centre for Mathematics and Computer Science, Amsterdam 1983.

[BK85] BERGSTRA, J.A. & J.W. KLOP, *Algebra of communicating processes with abstraction*, Theoret. Comput. Sci. 37 (1985), pp.77-121.

[BK84a] BERGSTRA, J.A. & J.W. KLOP, *Process algebra for synchronous communication*, Information and Control, Vol. 60, Nos. 1-3, pp.109-137, 1984.

[BK84b] BERGSTRA, J.A. & J.W. KLOP, *Verification of an alternating bit protocol by means of process algebra*, Report CS-R8404, Centre for Mathematics and Computer Science, Amsterdam 1984. Also in: Math. Methods of Spec. and Synthesis of Software Systems '85 (Eds. W. Bibel and K.P. Jantke), Math. *Research 31, Akademie-Verlag Berlin, 1986, pp.9-23.

[BK84c] BERGSTRA, J.A. & J.W. KLOP, *A complete inference system for regular processes with silent moves*, Report CS-R8420, Centre for Mathematics and Computer Science, Amsterdam 1984. To be published in the Proceedings of the Logic Colloquium in Hull, 1986 (eds. J. Truss and F. Drake). North-Holland.

[BK86a] BERGSTRA, J.A. & J.W. KLOP, *Algebra of Communicating Processes*, in: CWI Monographs I, Proceedings of the CWI Symposium Mathematics and Computer Science (eds. J.W. de Bakker, M. Hazewinkel and J.K. Lenstra). North-Holland, Amsterdam 1986, pp.89-138.

[BK86b] BERGSTRA, J.A. & J.W. KLOP, *Process algebra: specification and verification in bisimulation semantics*, in: CWI Monograph 4, Proceedings of the CWI Symposium Mathematics and Computer Science II (eds. M. Hazewinkel, J.K. Lenstra & L.G.L.T. Meertens), North-Holland, Amsterdam 1986, pp.61-94.

[BKO86] BERGSTRA, J.A., J.W. KLOP & E.-R. OLDEROG, *Failures without chaos: a new process semantics for fair abstraction*, in: Proceedings of the IFIP Working Conference on Formal Description of Programming Concepts, Gl. Avernaes 1986 (M. Wirsing, Ed.), North-Holland, Amsterdam 1987, pp.77-101.

[Br83] BROOKES, S.D., *On the relationship of CCS and CSP*, in: Proc. 10th Int. Colloq. Automat. Lang. & Programming, Barcelona (J.Díaz, Ed.), Lecture Notes in Computer Science No. 154, pp.83-96, Springer Verlag, New York/Berlin, 1983.

[BHR84] BROOKES, S., C. HOARE & W. ROSCOE, *A Theory of Communicating Sequential Processes*, J.Assoc. Comput. Mach. 31, No.3, pp.560-599, 1984.

[DH84] DE NICOLA, R. & M.C.B. HENNESSY, *Testing equivalences for processes*, Theoret. Comput. Sci. Vol.34, Nrs.1,2, pp. 83-133, 1984.

[He83] HENNESSY, M., *Synchronous and Asynchronous Experiments on Processes*, Information and Control,

Vol. 59, Nos. 1-3, pp.36-83, 1983.

[HP79] HENNESSY, M. & G.D. PLOTKIN, *Full abstraction for a simple programming language*, in: Proceedings 8th Mathematical Foundations of Computer Science (J. Becvar, Ed.), Lecture Notes in Computer Science No.74, Springer-Verlag, Berlin/New York, 1979, pp.108-120.

[Ho78] HOARE, C.A.R., *Communicating sequential processes*, Comm. ACM 21, pp.666-667, 1978.

[Ho80] HOARE, C.A.R., *A model for communicating sequential processes*, in: On the Construction of Programs (Eds.: R.M. McKeag and A.M. McNaughton), pp.229-243, Cambridge Univ. Press, London/New York, 1980.

[Mi77] MILNER, R., *Fully abstract models of typed λ-calculi*, Theoret. Comput. Sci. 4 (1977), pp.1-22.

[Mi80] MILNER, R., *A Calculus of Communicating Systems*, Lecture Notes in Computer Science No.92, Springer Verlag, New York/Berlin 1980.

[Mi83] MILNER, R., *Calculi for synchrony and asynchrony*, Theoret. Comput. Sci. 25 (1983), pp.267-310.

[OH83] OLDEROG, E.R. & C.A.R. HOARE, *Specification-oriented semantics for communicating processes*, in: Proc. 10th Int. Colloq. Automat. Lang. & Programming, Barcelona, Ed. J. Díaz, pp.561-572, Lecture Notes in Computer Science No. 154, Springer Verlag New York/Berlin, 1983; expanded version, Acta Informatica 23 (1986), pp.9-66.

[Pa83] PARK, D.M.R., *Concurrency and automata on infinite sequences*, in: Proc. 5th GI (Gesellschaft für Informatik) Conference, Springer LNCS 104, 1981.

[Pl77] PLOTKIN, G.D., *LCF considered as a programming language*, Theoret. Comput. Sci. 5 (1977), pp.223-255.

[RB81] ROUNDS,W.C. & S.D. BROOKES, *Possible futures, acceptances, refusals, and communicating processes*, in: Proceedings of 22nd IEEE Synposium on Foundations of Computer Science, Nashville, Tennessee (IEEE Computer Society Press, 1981) pp.140-149.

[Wi83] WINSKEL, G., *Synchronisation trees*, in: Proc. 10th ICALP, Barcelona (ed. J. Díaz), Springer LNCS 154, pp.695-711, 1983. Expanded version in Theoret. Comput. Sci. 34 (1984), pp.33-82.