



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

Department of Computer Science

Report CS-R8523

September

*Bibliotheek
Centrum voor Wiskunde en Informatica
Amsterdam*

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

J.A. Bergstra*

Universiteit van Amsterdam
Rijksuniversiteit Utrecht

J.W. Klop*

Centrum voor Wiskunde en Informatica
Amsterdam

E.-R. Olderog

Christian-Albrechts Universität
Kiel

Readiness and failure semantics are studied in the setting of ACP (algebra of communicating processes). A model of process graphs modulo readiness resp. 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 Milner's CCS and Hoare's CSP. 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.

1980 Mathematics Subject Classification: 68B10, 68C01, 68D25, 68F20.

1982 CR Categories: F.1.1, F.1.2, F.3.2, F.4.3.

Key Words & Phrases: process algebra, concurrency, readiness semantics, failure semantics, bisimulation semantics,

Note: This report will be submitted for publication elsewhere.

INTRODUCTION

This paper is concerned with the *failure semantics* for communicating processes as introduced in BROOKES, HOARE & ROSCOE [9] (see also ROUNDS & BROOKES [18]). 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 trace (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 trace semantics enriched by "local branching information".

Two further semantic models of processes will play an auxiliary rôle in our paper: Milner's model based on the notion of *observational equivalence* [14] or *bisimulation* (see PARK [17]) and the *readiness semantics* described in OLDEROG & HOARE [16]. 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 in BROOKES, HOARE & ROSCOE [9] and BROOKES [8] in the context of CSP (Communicating Sequential Processes, see [12, 13]) is the work of DE NICOLA & HENNESSY [10] 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 [10]

1. *) The research of J.A. Bergstra and J.W. Klop is partially supported by ESPRIT project 432: Meteor.

2. This report will be submitted for publication elsewhere.

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 [8].

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 [1-7].¹ 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 [15] and WINSKEL [19].) 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_r and displayed in Table 1. Note that ACP_r is purely equational and, for a finite alphabet A 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_r yield readiness semantics, and a "saturation" axiom S which when added to $ACP_r + R1,2$ yields failure semantics. ACP_r 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 $\equiv, \equiv_R, \equiv_F$ (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 BROOKES [8], and they are derivable from the axioms given in DE NICOLA & HENNESSY [10] (see our comparison in Remark 7.2.3). The definitions of $\equiv, \equiv_R, \equiv_F$ are also standard. What seems new in our treatment is the strategy of the completeness proofs by means of a decomposition of $\equiv, \equiv_R, \equiv_F$ on process graphs in a small number of very simple *process graph transformations* (Section 3).

So we obtain a "graph model" for ACP_r 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 BROOKES, HOARE & ROSCOE [9]. The graph model and the failure model are shown to be isomorphic. Section 6 connects our work with CCS and CSP by formulating their communication concepts as a restriction of the general communication format in ACP_r . This serves as a preparation for Section 7 where we show that under this restriction failure equivalence is the *maximal trace respecting congruence*. The proof of this result uses the readiness semantics as a "stepping stone" towards failure equivalence. Though the proof techniques resemble somewhat the notion of testing in DE NICOLA & HENNESSY [10] (see Remark 7.2.3), this simple characterisation of failure equivalence seems new.

The paper concludes in Section 8 with a digression in which processes under failure semantics are

1. (This paper is in principle self-contained, but the survey [6] will help the understanding. All papers [1-7] contain some material referred to in the sequel; the advised process for reading them would be, in the process notation of these papers:

$((6 \parallel 7) \cdot (2 + 3) \cdot 4) \parallel 5$. That is, [6] and [7] can be considered in parallel, taking the first part from [6], followed by a choice of [2] and [3], followed by [4] and [1]; [5] can be read rather independently.)

$x + y = y + x$	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
$a b = b a$	C1
$(a b) c = a (b c)$	C2
$\delta a = \delta$	C3
$x y = x y + y x + x y$	CM1
$a x = ax$	CM2
$(ax) y = a(x y)$	CM3
$(x + y) z = x z + y z$	CM4
$(ax) b = (a b)x$	CM5
$a (bx) = (a b)x$	CM6
$(ax) (by) = (a b)(x y)$	CM7
$(x + y) z = x z + y z$	CM8
$x (y + z) = x y + x z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$\partial_H(xy) = \partial_H(x) \cdot \partial_H(y)$	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

TABLE 1. Algebra of communicating processes with renaming. Here a, b range over the set $A_\delta (= A \cup \{\delta\})$ 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 $+, \cdot, ||, ||, |$ and the unary operations ∂_H, a_H where $H \subseteq A$. See Section 1.2 for further explanation.

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.

We conclude this introduction with a table of contents.

CONTENTS

1. THE DOMAIN H_δ OF FINITE ACYCLIC PROCESS GRAPHS

- 1.1. Finite acyclic process graphs in δ -normal form.
- 1.2. Operations on process graphs.

2. EQUIVALENCES ON PROCESS GRAPHS

- 2.1. Trace equivalence.
- 2.2. Ready equivalence and failure equivalence.
- 2.3. Bisimulation equivalence.
- 2.4. Comparing the equivalences.
- 2.5. Convexly saturated process graphs.

3. TRANSFORMATIONS ON PROCESS GRAPHS

- 3.1. The transformations double edge, sharing, cross and fork.
- 3.2. Connecting process graph equivalences with process graph transformations.

4. AXIOMATISING THE EQUIVALENCES ON PROCESS GRAPHS

- 4.1. The case without communication.
- 4.2. The case with communication: the graph model of ACP_r .

5. THE FAILURE MODEL OF ACP_r

- 5.1. The domain F of failure sets.
- 5.2. The failure model.

6. ACP_r WITH 1-1 COMMUNICATION

- 6.1. 1-1 communication.
- 6.2. Milner's parallel composition $\parallel_{\mathcal{M}}$ in CCS.
- 6.3. Hoare's parallel composition $\parallel_{\mathcal{H}}$ in CSP.

7. THE MAXIMAL TRACE RESPECTING CONGRUENCE

- 7.1. Preliminaries.
- 7.2. A characterisation of failure equivalence.

8. PROCESSES WITH RECURSION AND ABSTRACTION: BISIMULATION VERSUS FAILURE EQUIVALENCE

- 8.1. Preliminaries.
- 8.2. The inconsistency of failure semantics with KFAR.

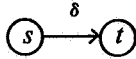
REFERENCES

1. THE DOMAIN \mathbb{H}_δ 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 with introducing a domain of process graphs (\mathbb{H}_δ) enriched with a number of operations $+, \cdot, \parallel, \lfloor _, \rfloor, \mid, \partial_H, a_H$ corresponding to the operators in ACP_r . It should be emphasized that this structure $\mathbb{H}_\delta(+, \cdot, \parallel, \lfloor _, \rfloor, \mid, \partial_H, a_H)$ is not yet a model of ACP_r ; it becomes so after dividing out a suitable equivalence on \mathbb{H}_δ (which of course should be a congruence w.r.t. the operations). For example, dividing out 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 readiness equivalence resp. failure equivalence (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 resp. failure semantics.

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 collection of *finite acyclic* process graphs over the alphabet $A_\delta = A \cup \{\delta\}$ consisting of actions $a, b, \dots \in A$ 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



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 $\circ s \xrightarrow{\delta} \circ t$ is an *ancestor* of $\circ s' \xrightarrow{\delta} \circ t'$ 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 begin 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 $\circ s \xrightarrow{\delta} \circ t$ in g consists of deleting this edge, creating a fresh node t' and adding the edge $\circ s \xrightarrow{\delta} \circ t'$

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 disconnected from the root, are discarded.

1.1.1 Example.

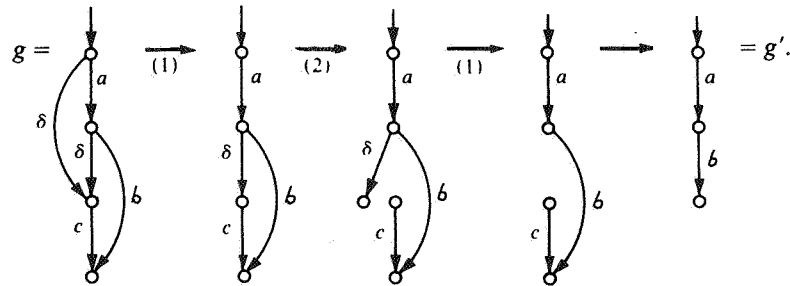
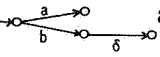
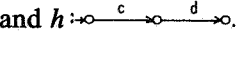


Fig. 1.

1.2. Operations on process graphs.

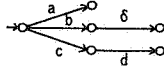
On \mathbb{H}_δ we define the operations $+$, \cdot , \parallel , \sqcup , ∂_H , as in [3,6], and moreover renaming operators a_H . For the sake of completeness we repeat the definitions briefly:

- (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 $\rightarrow_{\delta} \bigcirc \rightarrow \bigcirc$).
- (ii) the *product* $g \cdot h$ is obtained by appending h at all terminal nodes which are not terminal nodes of a δ -step;
- (iii) the *merge* $g \parallel h$ 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* $g \sqcup h$ is the subgraph of $g \parallel h$ where an initial step must be one from g ;
- (v) the *communication merge* $g | h$ is the subgraph of $g \parallel h$ 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 $H \subseteq A$ by δ , and taking the δ -normal form.
- (vii) the *renaming* $a_H(g)$ is the result of renaming all (labels of) steps in $H \subseteq A$ by a . We have renamings a_H for each $a \in A$.

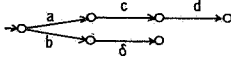
1.2.1 Example. Let g be  and h . Let the communication function

$|\cdot| : A_\delta \times A_\delta \rightarrow A_\delta$ be such that $a | c = e$ and $b | d = f$, all other communications equal δ . Then:

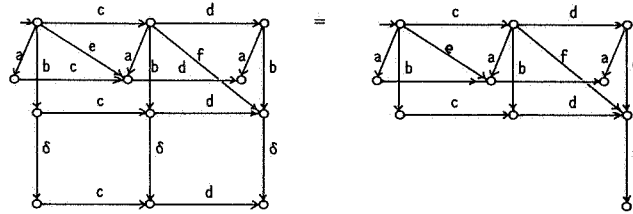
(i) $g + h =$



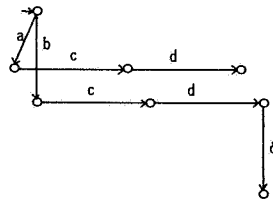
(ii) $g \cdot h =$



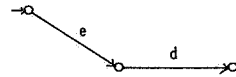
(iii) $g \parallel h =$ the δ -n.f. of



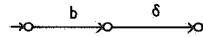
(iv) $g \sqcup h =$

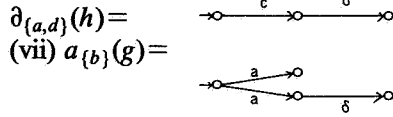


(v) $g | h =$



(vi) $\partial_{\{a,d\}}(g) =$





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 resp. failure equivalence presented in Section 4.

2.1. Trace equivalence

Let $g \in \mathbb{H}_\delta$. Then $\sigma \in A^*$ is a *trace* of g if there is a path in g from the root of g to a terminal node such that σ is the word formed by concatenating the labels in the consecutive steps in the path. The set of all traces of g is $\text{trace}(g)$.

Notation: if $g, h \in \mathbb{H}_\delta$ then $g \sim_{tr} h$ means: $\text{trace}(g) = \text{trace}(h)$.

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.

Let s_0 be the root of g . Then each path in g starting from s_0 and ending in a proper node (i.e. not of type (i)), determines a word (or *history*) $\sigma \in A^*$. The trivial path of zero steps determines the empty word ϵ . We call σ a *successful trace* if it is determined by a path from s_0 to a successful termination node. So σ is a trace as in 1.3 not ending in δ .

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: $\mathbb{R}[g]$.

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

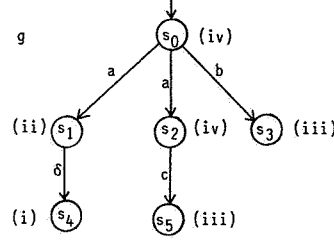
$$\mathbb{R}[g] = \{\sigma \mid \sigma \text{ is successful trace of } g\} \cup \{(\sigma, X) \mid (\sigma, X) \text{ is ready pair of } g\}$$

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

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

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

FIGURE 3



	$\mathcal{R}[g]$	$\mathcal{A}[g]$
s_0	$(\epsilon, \{a, b\})$	$[\epsilon, Y], Y \subseteq A - \{a, b\}$
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

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

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

(iii) Let $a\delta$ be the graph $\text{graph} \begin{array}{c} \circ \xrightarrow{a} \circ \xrightarrow{\delta} \circ \end{array}$. Then $\mathcal{R}[a\delta] = \{(\epsilon, \{a\}), (a, \emptyset)\}$ and $\mathcal{A}[a\delta] = \{[\epsilon, Y] | Y \subseteq A - \{a\}\} \cup \{[a, Z] | Z \subseteq A\}$.

2.2.3 Definition. Let $g, h \in \mathbb{H}_\delta$. Then:

$$g \equiv_{\mathcal{R}} h \text{ if } \mathcal{R}[g] = \mathcal{R}[h]$$

$$g \equiv_{\mathcal{A}} h \text{ if } \mathcal{A}[g] = \mathcal{A}[h].$$

In words: g, h are *ready equivalent* resp. *failure equivalent*.

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 $\text{ROOT}(g)$, $\text{ROOT}(h)$ denote the root of g, h and $\text{NODES}(g)$, $\text{NODES}(h)$ denote the set of nodes of g, h .

Then $R \subseteq \text{NODES}(g) \times \text{NODES}(h)$ is a *bisimulation from g to h* if:

(i) $(\text{ROOT}(g), \text{ROOT}(h)) \in R$

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

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

Notation: $g \stackrel{u}{\leftrightarrow} h$ (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 \Leftrightarrow : for $g, h \in \mathbb{H}_\delta$ we have

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

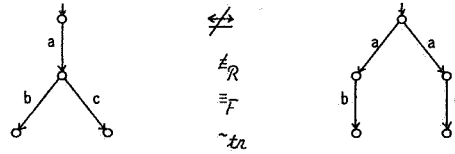
and in general none of these implications can be reversed as some of the following examples (2.4.1) show. Lemma 2.5.5 states a sufficient condition for reversing the second implication.

In the sequel we will prove (Proposition 4.2.3) that $\equiv_{\mathcal{R}}$ and $\equiv_{\mathcal{F}}$ are *congruences* w.r.t. the operations defined above in 1.2. Also \Leftrightarrow is a congruence; see [3], Theorem 2.5 for the more complicated situation where τ -steps are present. Trace equivalence however is *not* a congruence w.r.t. these operations, as the following example shows.

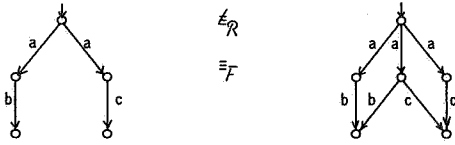
EXAMPLE. Let $C[\xi]$ be the context $\partial_{\{b,c\}}(\xi \parallel c)$, and let a, b, b^0, c, c^0 be atoms with communications $b \mid b = b^0$, $c \mid c = c^0$ and all other communications resulting in δ . Consider the trace equivalent processes $a(b+c)$ and $ab+ac$. Then $C[a(b+c)] = ac^0 \neq a\delta + ac^0 = C[ab+ac]$.

EXAMPLES.

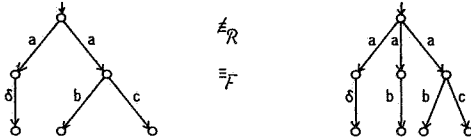
(i)



(ii)



(iii)



(iv)

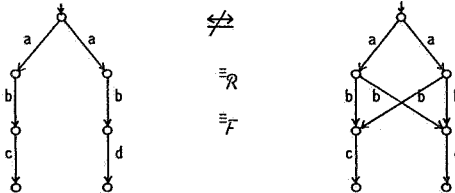


FIGURE 4

2.5. Convexly saturated process graphs.

Following BROOKES [8] and DE NICOLA & HENNESSY [10] we introduce:

2.5.1. DEFINITION. $\mathcal{X} \subseteq \mathcal{P}(A)$ is *convex* if

- (i) $X, Y \in \mathcal{X} \Rightarrow X \cup Y \in \mathcal{X}$
- (ii) $X, Y \in \mathcal{X}, X \subseteq Z \subseteq Y \Rightarrow Z \in \mathcal{X}$.

(In particular, $\emptyset \subseteq \mathcal{P}(A)$ is convex.)

DEFINITION 2.5.2. (i) Let $g \in \mathbb{H}_\delta$ and $\sigma \in A^*$. Then $g \mid \sigma = \{X \mid \exists (\sigma, X) \in \mathcal{R}[g]\}$.

(ii) g is *convexly saturated* (or just 'convex' or 'saturated') if $g \mid \sigma$ is convex, for all $\sigma \in A^*$.

EXAMPLE 2.5.3. In Figure 5, g_1, g_2 are not convexly saturated, but their 'convex saturations' g'_1, g'_2 are.

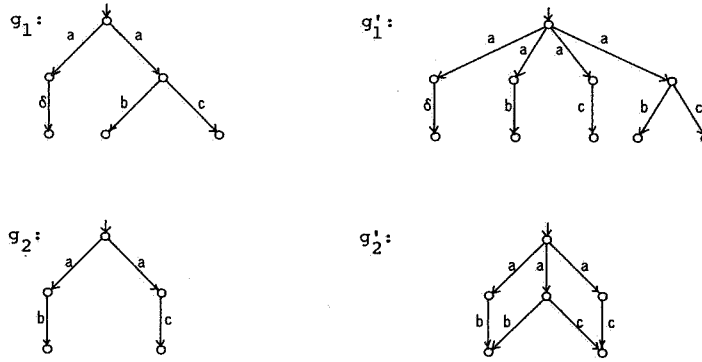


FIGURE 5.

PROPOSITION 2.5.4. Let $\mathcal{X} \subseteq \mathcal{P}(A)$ be convex, and suppose $Y \notin \mathcal{X}$, $Y \subseteq \bigcup \mathcal{X}$. Then for all $X \in \mathcal{X}$: $\bar{Y} \subseteq \bar{X}$.

PROOF. First we note that a convex \mathcal{X} is closed under taking supersets, when relativised to $\bigcup \mathcal{X}$. That is, if \mathcal{X} is convex, then:

$$X \in \mathcal{X}, Y \supseteq X \Rightarrow Y \cap \bigcup \mathcal{X} \in \mathcal{X} \quad (*)$$

Now suppose for a proof by contradiction that $\bar{Y} \not\subseteq \bar{X}$ for some $X \in \mathcal{X}$. Then $X \subseteq Y$. Hence, by (*) and the assumption $Y \subseteq \bigcup \mathcal{X}$, we have $Y \in \mathcal{X}$; contradiction. \square

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

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

PROOF. Only to prove (\Leftarrow). So, we suppose $g \not\equiv_{\mathcal{R}} h$ and we want to prove $g \not\equiv_{\mathcal{S}} h$.

We may suppose further that g, h have the same trace set, otherwise $g \not\equiv_{\mathcal{S}} h$ 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 $[\sigma, \bar{X}] \in \mathcal{F}[g]$. Now consider $h \mid \sigma$, which is by assumption convex. Since $g \simeq_{\mathcal{R}} h$, we have $X \subseteq \bigcup h \mid \sigma$. Further, $(\sigma, X) \notin \mathcal{R}[h]$ entails $X \not\subseteq h \mid \sigma = \{X_i \mid i \in I\}$. So, by Proposition 2.5.4: $X \not\subseteq X_i$, for all i . But then

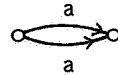
$[\sigma, \bar{X}] \notin \mathcal{H}$ and we have $g \not\equiv h$. \square

3. TRANSFORMATIONS ON PROCESS GRAPHS

We now introduce four *elementary transformations* on process graphs $\in \mathcal{H}_\delta$ with the following property: the first two of them generate, when applied on $g \in \mathcal{H}_\delta$, 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*



($a \in A$) one of the edges.

Notation: $g \xRightarrow{[i]} h$.

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

[iii] **cross**. If $g \in \mathcal{H}_\delta$ contains a part as in Figure 6(i), edges as in Figure 6(ii) may be inserted:



FIGURE 6.

Notation: $g \xRightarrow{[iii]} h$.

[iv] **fork**. Let $g \in \mathcal{H}_\delta$ contain a part as in Figure 7 (i) where all successor steps b_1, \dots, b_n of the left a -step are displayed. Then a part as indicated in Figure 7 (ii) may be inserted. Notation: $g \xRightarrow{[iv]} h$.

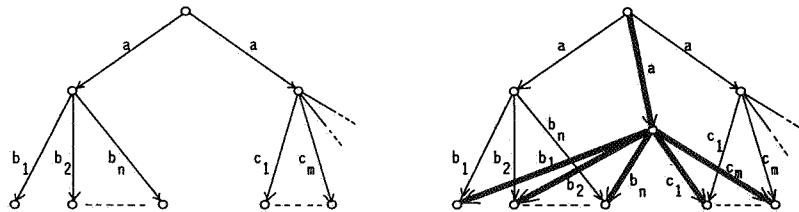


FIGURE 7.

Here it is not required that all steps $b_1, \dots, b_n, c_1, \dots, c_m$ 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.

NOTATION 3.1.1.

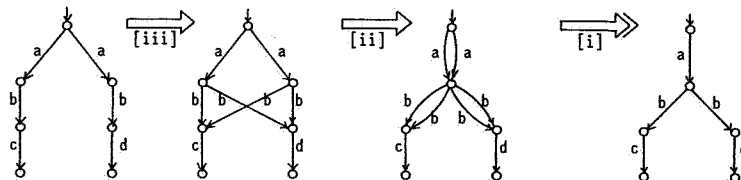
$\xRightarrow{[i]}$ is $\xRightarrow{[i]} \cup \dots \cup \xRightarrow{[iv]}$;

\Rightarrow is the transitive reflexive closure of \Rightarrow ;

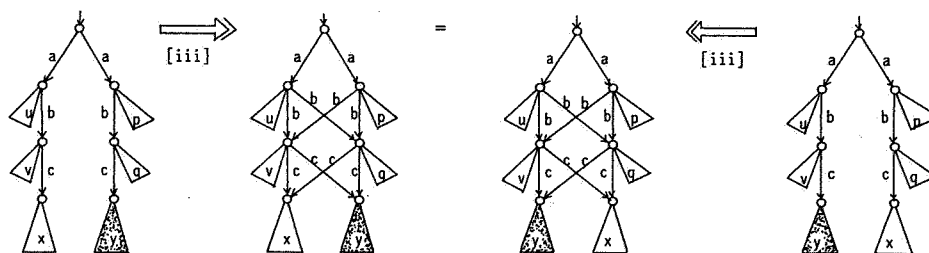
\Leftrightarrow is the equivalence relation generated by \Rightarrow .

EXAMPLE 3.1.2.

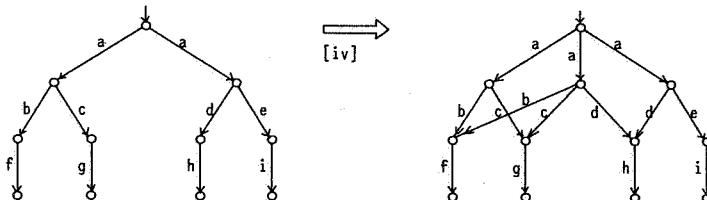
(i)



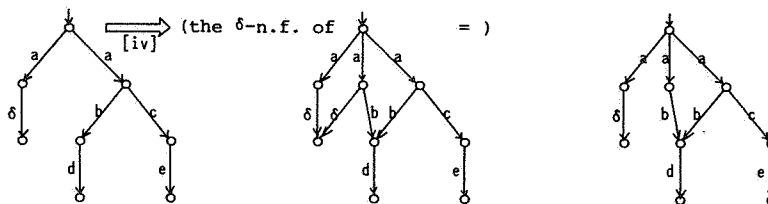
(ii) Note how \Rightarrow enables one to switch subgraphs x, y at the end of paths with the same history (abc in the following example):



(iii)



(iv)



(v)

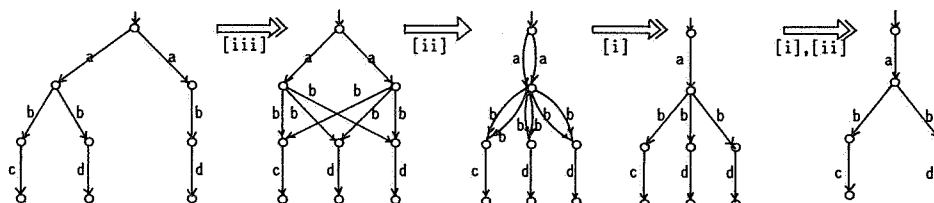


FIGURE 8.

3.2. Connecting process graph equivalences with process graph transformations

3.2.1 PROPOSITION Let $g, h \in \mathbb{H}_\delta$. Then:

- (i) $g \xrightarrow{[i][ii][iii]} h \Rightarrow g \equiv_{\mathcal{R}} h$
- (ii) $g \xrightarrow{[i]-[iv]} h \Rightarrow g \equiv_{\mathcal{R}} h$.

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 7 (ii)).

Case 1. Let $(\sigma a, \{b_1, \dots, b_n\})$ 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 $(\sigma a, \{b_1, \dots, b_n, c_1, \dots, c_m\})$. 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 $\mathcal{R}[g]$ which consists of successful traces, is invariant. \square

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}_δ :

DEFINITION. 3.2.2. (i) Let $g \in \mathbb{H}_\delta$ have ready set $\mathcal{R}[g]$. Then

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

is the process graph with $\mathcal{R}[g] \cup \{\bigcirc\}$ as set of nodes, with $(\epsilon, X) \in \mathcal{R}[g]$ as root, and with edges given by

$$(\sigma, \{a\} \cup X) \xrightarrow{a} (\sigma a, Y)$$

$$(\sigma, \{a\} \cup X) \xrightarrow{a} \sigma a$$

$$(\sigma, \emptyset) \xrightarrow{\delta} \bigcirc$$

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

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

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

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

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

3.2.3 Example. Let g be as in Figure 9(i). Then $\mathcal{R}(g), \mathcal{R}(g)$ are as in Figure 9(ii), (iii):

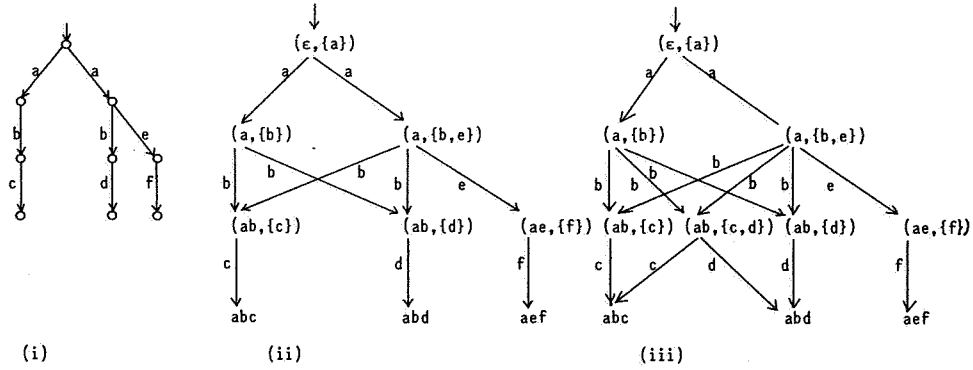


FIGURE 9.

3.2.4 PROPOSITION

- (i) $g \xleftrightarrow[i]{[i]-[iii]} \mathcal{R}(g)$
- (ii) $g \xleftrightarrow{} \mathcal{R}(g)$
- (iii) $g \equiv_{\mathcal{R}} \mathcal{R}(g)$
- (iv) $g \equiv_{\mathcal{R}} \mathcal{R}(g)$
- (v) $\mathcal{R}(\mathcal{R}(g)) = \mathcal{R}(g)$.
- (vi) $g \equiv_{\mathcal{R}} h \Rightarrow \mathcal{R}(g) = \mathcal{R}(h)$
- (vii) $g \equiv_{\mathcal{R}} h \Rightarrow \mathcal{R}(g) = \mathcal{R}(h)$.

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 w.r.t. $\Rightarrow_{[i]}, \Rightarrow_{[ii]}$. (This does not make further applications of $\Rightarrow_{[iii]}$ possible.) Call the result of the procedure (1)-(3): $\mathcal{R}(g)$. **Claim.** $\mathcal{R}(g) = \mathcal{R}(g)$.

Proof of the claim. If s is a non-terminal node of $\mathcal{R}(g)$, 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) resp. σ_s depends uniquely from s , by (1) of the procedure. Further, the ready set of g coincides with that of $\mathcal{R}(g)$, by Proposition 3.2.1 (i).

Hence $\phi: s \mapsto (\sigma_s, X_s)$ resp. σ_s 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 $\mathcal{R}(g)$ with $(\sigma_s, X_s) = (\sigma_{s'}, X_{s'})$ and $s \neq s'$, then by (2) of the construction of $\mathcal{R}(g)$ there are 'crosses' between each two steps a, a from s, s' respectively (see Figure 10):

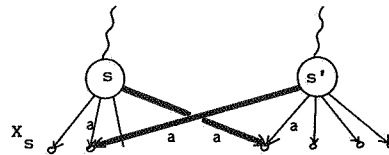


FIGURE 10.

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: $s \xrightarrow{a} t$ iff

$$\phi(s) = (\sigma_s, X_s) = (\sigma_s, \{a\} \cup X'_s), \phi(t) = (\sigma_s a, Y)$$

for some X_s, X'_s, Y , iff

$$\phi(s) \xrightarrow{a} \phi(t).$$

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 $\begin{smallmatrix} \longleftrightarrow \\ [i], [ii] \end{smallmatrix}$ nothing new is added and then taking the normal form w.r.t. [i], [ii].

Hence it follows from (i) that $g \iff \mathcal{F}(g)$. (iii), (iv): 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. \square

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

(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): $\mathcal{F}(g) = \mathcal{F}(h)$. \square

3.2.5 COROLLARY. Let $g, h \in \mathbb{H}_\delta$. Then:

- (i) $g \iff h$ iff $g \begin{smallmatrix} \longleftrightarrow \\ [i], [ii] \end{smallmatrix} h$
- (ii) $g \equiv_{\mathcal{R}} h$ iff $g \begin{smallmatrix} \longleftrightarrow \\ [i] - [iii] \end{smallmatrix} h$
- (iii) $g \equiv_{\mathcal{F}} h$ iff $g \longleftrightarrow h$

PROOF. (i) is (essentially) proved in [2] (Appendix) and also in [3] (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 right to left follows from Proposition 3.2.1(i). The other direction follows from Proposition 3.2.4 (i), (vi).

(iii) Similar as (ii). \square

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}_δ 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 $\parallel, \sqcup, |, \partial_H$ 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 $\mathbb{H}_\delta(+, \cdot) / \equiv_{\mathcal{R}}, \mathbb{H}_\delta(+, \cdot) / \equiv_{\mathcal{F}}$. These are the structures which we will now axiomatise.

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

REMARK. (i) The axioms R1, 2 and S (R for *readiness*, S for *saturation*) which are specific for failure equivalence, appear already in BROOKES [8] in a slightly different form. BROOKES [8] considers also τ -steps and presents as laws valid for failure equivalence in Proposition 1.3.6:

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

(here $\mu \in A_\delta \cup \{\tau\}$; x, y, u, v are arbitrary processes), and in Proposition A.3 in [8]:

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

Clearly 1, 2 imply R1 in Table 2 below; 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

$$\begin{aligned} ax + ay &= ax + ay + a(x + y) \\ ax + a(x + y + z) &= ax + a(x + y) + a(x + y + z) \end{aligned}$$

(where $a \in A_\delta$) which is equivalent to axiom S in Table 2.

(ii) For a comparison with the work of the DE NICOLA & HENNESSY [10], see Remark 7.2.3.

4.1.1. $BPA_\delta + R1, 2 + S$ is the axiom system displayed in Table 2.

$x + y = y + x$	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

TABLE 2.

Here a, b vary over $A \cup \{\delta\}$; x, y, z, u, v are variables for processes.

4.1.2 Connecting terms with process graphs.

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

$$\text{graph} : \text{Ter}(BPA_\delta) \rightarrow \mathbb{H}_\delta$$

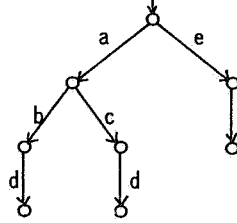
$$\text{ter} : \mathbb{H}_\delta \rightarrow \text{Ter}(BPA_\delta)$$

as follows. $\text{graph}(T)$ is the process graph obtained by first normalizing T w.r.t. A4,6,7 in Table 2 and second interpreting $a, +, \cdot$ as the corresponding 'one edge graphs' and operators $+, \cdot$ on \mathbb{H}_δ .

EXAMPLE:

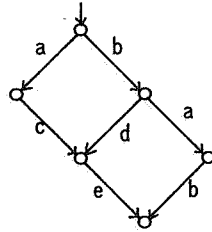
$$\text{graph}(a(b+c+\delta)d + \delta e + e\delta) =$$

$$\text{graph}(a(bd + cd) + e\delta) =$$

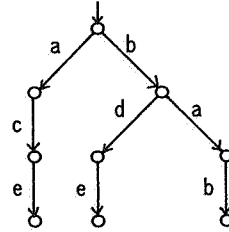


Further, to define $\text{ter}(g)$ we first define $\text{tree}(g)$ as the tree obtained from g by 'unsharing'.

EXAMPLE: if g is as in Figure 11(i), then $\text{tree}(g)$ is as in Figure 11(ii):



(i)



(ii)

FIGURE 11.

Now we define $\text{ter}(g)$ as the term corresponding in the obvious way with $\text{tree}(g)$.

Example: if g is as above, then $\text{ter}(g) = ace + b(de + ab)$.

4.1.3. REMARK. (i) Note that the detour via $\text{tree}(g)$ is necessary since e.g. the graph g above is for no term T equal to $\text{graph}(T)$.

(ii) Further we note that ter, graph are 'almost' inverse to each other:

$$BPA_\delta \vdash (\text{ter} \circ \text{graph})(T) = T$$

$$(\text{graph} \circ \text{ter})(g) \Leftrightarrow g$$

where \Leftrightarrow (bisimulation) coincides with $\langle\langle \cdot \rangle\rangle_{[\nu], [ii]}$.

4.1.4. TRANSFER LEMMA. Let $g, h \in H_\delta$ be such that $g \Rightarrow h$. Then

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

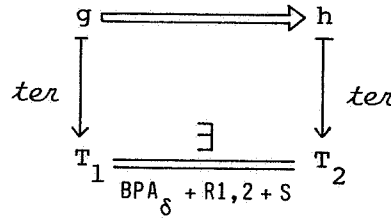


FIGURE 12.

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

A transformation $g \xRightarrow{[ii]} h$ is invisible on the level of terms, i.e. $ter(g)$ and $ter(h)$ are identical terms. Next consider a transformation $g \xRightarrow{[iii]} h$, which consists of adding two edges in g as in Figure 13:

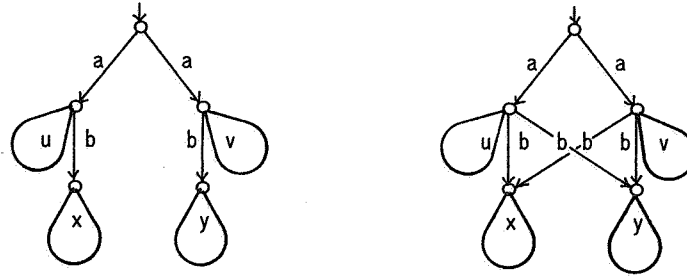


FIGURE 13.

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.

Finally, a transformation $g \xRightarrow{[iv]} h$ (see also Figure 7) translates into an application of axiom S in Table 2. \square

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

PROOF. We prove (ii); the proof of (i) is similar, noting that the proof of Lemma 4.1.4 shows that \Rightarrow is transferred to applications of axioms in $BPA_\delta + R1, 2$.

[i]–[iii] Checking the soundness (\Rightarrow) is routine and will not be done here.

The completeness (\Leftarrow): suppose $graph(T_1) \equiv_{\mathcal{A}} graph(T_2)$. 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):

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

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) \simeq defines isomorphism between algebras.

4.1.6. COROLLARY. (i) $\mathbb{H}_\delta(+, \cdot) / \equiv_{\mathfrak{A}} \simeq I(BPA_\delta + R1, 2)$

(ii) $\mathbb{H}_\delta(+, \cdot) / \equiv_{\mathfrak{F}} \simeq I(BPA_\delta + R1, 2 + S)$. \square

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 $\parallel, \underline{\parallel}, |, \partial_H, a_H$ on \mathbb{H}_δ 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 3, which presents the axiom system $ACP_r + R1, 2 + S$, and which extends our earlier axiom system $BPA_\delta + R1, 2 + S$ in Table 2.

As before, in Table 3 on the next page, 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 $\equiv_{\mathfrak{F}}$, called the *graph model* for $ACP_r + R1, 2 + S$. To this end we have first to prove that $\equiv_{\mathfrak{F}}$ is a *congruence* w.r.t. also the new operators $\parallel, \underline{\parallel}, |, \partial_H, a_H$. Once we have this, and knowing from [3,6] (after leaving out all reference to τ -steps) that there is the isomorphism

$$I(ACP_r) \simeq \mathbb{H}_\delta(+, \cdot, \parallel, \underline{\parallel}, |, \partial_H, a_H) / \Leftrightarrow$$

where \Leftrightarrow is bisimulation (which coincides with $\langle\langle \xrightarrow{\quad} \rangle\rangle_{[i], [u]}$; Corollary 3.2.5(i)), the derived isomorphism is a consequence from some general facts which we will state now (in 4.2.1).

$x + y = y + x$	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
$a b = b a$	C1
$(a b) c = a (b c)$	C2
$\delta a = \delta$	C3
$x \parallel y = x \parallel y + y \parallel x + x y$	CM1
$a \parallel x = ax$	CM2
$(ax) \parallel y = a(x \parallel y)$	CM3
$(x + y) \parallel z = x \parallel z + y \parallel z$	CM4
$(ax) b = (a b)x$	CM5
$a (bx) = (a b)x$	CM6
$(ax) (by) = (a b)(x \parallel y)$	CM7
$(x + y) z = x z + y z$	CM8
$x (y + z) = x y + x z$	CM9
$\partial_H(a) = a$ if $a \notin H$	D1
$\partial_H(a) = \delta$ if $a \in H$	D2
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$\partial_H(xy) = \partial_H(x) \cdot \partial_H(y)$	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

$ACP_r + R1,2 + S$

TABLE 3.

4.2.1. General intermezzo.

I.

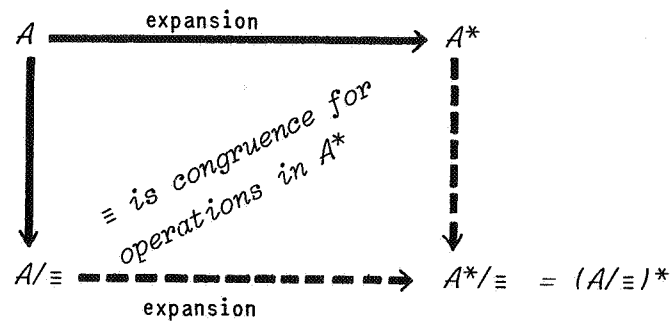


FIGURE 14.

Let \mathcal{Q} be an algebra which on the one hand can be *expanded* to \mathcal{Q}^* (i.e. enriched with new functions; the domain is invariant) and on the other hand can be factored out via \equiv , a congruence on \mathcal{Q} , to \mathcal{Q}/\equiv . Suppose moreover that \equiv is also a congruence on \mathcal{Q}^* . (See Figure 14.)

Then this expansion and factorisation are compatible (or commuting): \mathcal{Q}^*/\equiv equals $(\mathcal{Q}/\equiv)^*$.

II. Now let $\mathcal{Q}, \mathcal{Q}^*, \mathcal{Q}/\equiv$ (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*.

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

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

$$\mathcal{Q}^*/\equiv = (\mathcal{Q}/\equiv)^* \simeq I(\Sigma \cup \Delta, E \cup D \cup F)$$

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

4.2.2. THEOREM. Let the initial algebras $I(BPA_\delta)$ etc. as in Figure 15 (ii) of the axiom systems BPA_δ etc. as in Figure 15(i) be given. Furthermore, consider the graph models as in Figure 15(iii).

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

$$I(ACP_r + R1,2 + S) \simeq \mathbb{H}_\delta(+, \cdot, \parallel, \perp, |, \partial_H, a_H) / \equiv_{\mathcal{F}}$$

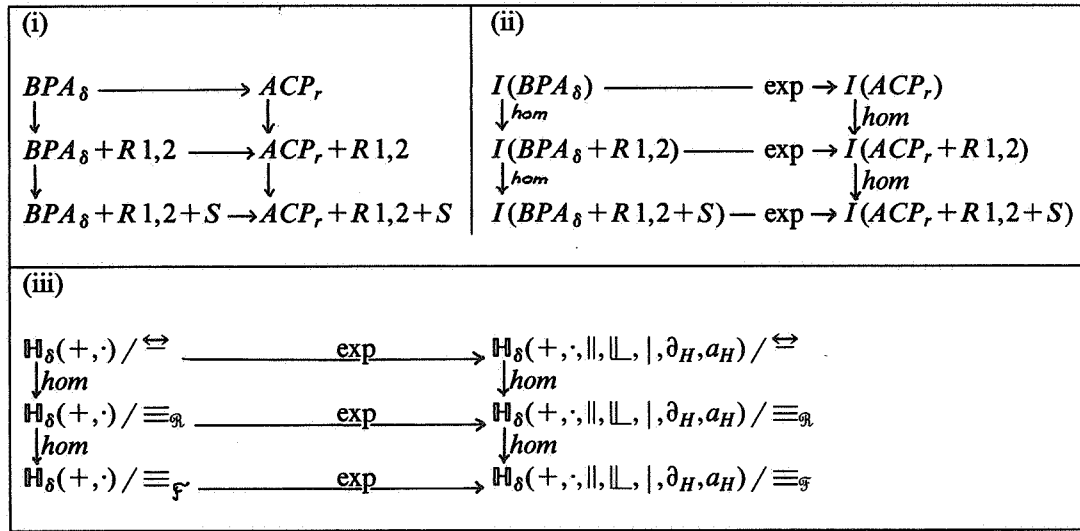


FIGURE 15.

PROOF. Consider e.g.

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

the corresponding initial algebras $I(BPA_\delta)$ etc. and the (by position in the diagram above) corresponding graph models

$$\begin{array}{ccc} \mathbb{H}_\delta(+, \cdot) / \simeq & \xrightarrow{\text{exp}} & \mathbb{H}_\delta(+, \cdot, \parallel, \llbracket, \mid, \partial_H, a_H) / \simeq \\ \text{hom} \downarrow & & \\ \mathbb{H}_\delta(+, \cdot) / \equiv_{\mathcal{F}} & & \end{array}$$

By Corollary 4.1.6(ii) we have $I(BPA_\delta + R1, 2 + S) \simeq \mathbb{H}_\delta(+, \cdot) / \equiv_{\mathcal{F}}$, and by results in [3,6] we have $I(BPA_\delta) \simeq \mathbb{H}_\delta(+, \cdot) / \simeq$ and $I(ACP_r) \simeq \mathbb{H}_\delta(+, \cdot, \parallel, \llbracket, \mid, \partial_H, a_H) / \simeq$.

Therefore, by 4.2.1.III, it suffices to prove that $\equiv_{\mathcal{F}}$ is a congruence w.r.t. the 'new' operators on \mathbb{H}_δ in order to conclude that $I(ACP_r + R1, 2 + S) \simeq \mathbb{H}_\delta(+, \cdot, \parallel, \llbracket, \mid, \partial_H, a_H) / \equiv_{\mathcal{F}}$. This is proved in the next proposition. \square

4.2.3. PROPOSITION. (i) Failure equivalence is a congruence w.r.t. the operators $\parallel, \llbracket, \mid, \partial_H, a_H$ on \mathbb{H}_δ .
(ii) The same holds for ready equivalence.

PROOF. (i) *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 $g \Rightarrow h \Rightarrow \partial_H(g) \equiv_{\mathcal{F}} \partial_H(h)$. The cases that \Rightarrow is $\Rightarrow_{[i]}$ or $\Rightarrow_{[ii]}$ present no problem. $\Rightarrow_{[iii]}$: it is easy to verify that

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

$\Rightarrow_{[iv]}$: 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 not to have the following graphs g, h as failure equivalent:



since $\partial_{\{b\}}$ would yield a trace $a\delta$ in h but not in g .)

THE CASE OF \parallel . It suffices to prove:

$$g \Rightarrow g' \Rightarrow g \parallel h \equiv_{\mathcal{F}} g' \parallel h.$$

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

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

PROOF OF (1): Due to the construction of a merge as a cartesian product with diagonal edges for communications (Figure 16 (i)), it is 'geometrically' clear (see Figure 16 (ii)) 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 $g \parallel h \Rightarrow_{[iii]} g' \parallel h$.

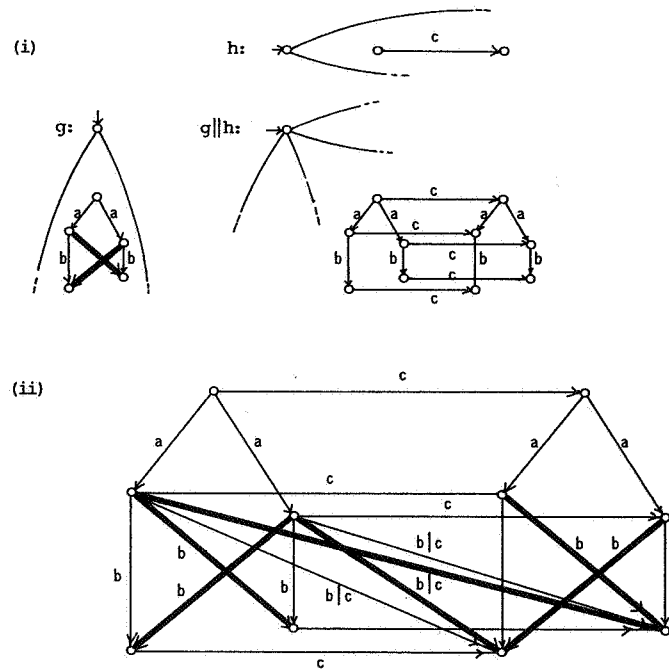


Figure 16

PROOF OF (2). Under the assumption $g \Rightarrow_{[iv]} g'$ we now prove $g || h \equiv_g g' || h$ directly from the definition of \equiv_g . So consider the addition in g of a fork which connects all successors of s_1 (see Figure 17) 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 $g' || h$ 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_1, t) by some consideration of the outgoing edges of (s_1, t) and (s_2, t) . The precise verification is omitted here.

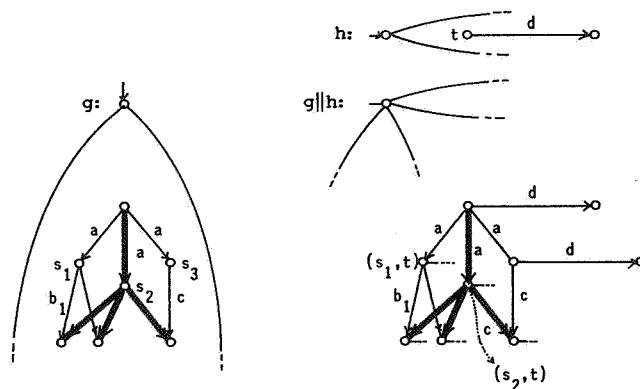


FIGURE 17.

(ii) as (i), but simpler. \square

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}_δ , and it was shown to be a congruence w.r.t. the operators of ACP_r in \mathbb{H}_δ . The quotient $\mathbb{H}_\delta / \equiv_{\mathcal{F}}$ 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 $\equiv_{\mathcal{F}}$ in the sense of

$$I(ACP_r + R1,2 + S) \simeq \mathbb{H}_\delta / \equiv_{\mathcal{F}}.$$

Here $\mathbb{H}_\delta / \equiv_{\mathcal{F}}$ is short for $\mathbb{H}_\delta(+, \cdot, \parallel, \lfloor, \rfloor, \partial_H, a_H) / \equiv_{\mathcal{F}}$. In this section we will provide an *explicit representation* of the quotient structure $\mathbb{H}_\delta(+, \cdot, \parallel, \lfloor, \rfloor, \partial_H, a_H) / \equiv_{\mathcal{F}}$, called the *failure model* of ACP_r . The model will shed more light into the structure of failures, and it will link our definitions with the original work on failures in BROOKES, HOARE & ROSCOE [9].

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

$$F \subseteq A^+ \cup (A^* \times \mathcal{P}(A))$$

which satisfy the following closure properties:

- (i) $[\epsilon, \emptyset] \in F$
- (ii) $[\sigma_1 \sigma_2, \emptyset] \in F$ implies $[\sigma_1, \emptyset] \in F$
- (iii) $X \subseteq Y$ and $[\sigma, Y] \in F$ imply $[\sigma, X] \in F$
- (iv) $[\sigma, X] \in F$ and $[\sigma, X \cup \{a\}] \notin F$ imply $\sigma a \in F$ or $[\sigma a, \emptyset] \in F$
- (v) $\sigma a \in F$ implies $[\sigma, \emptyset] \in F$.

For failure sets $F \subseteq A^* \times \mathcal{P}(A)$ 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 [9].

Next we define the operations $+, \cdot, \parallel, \lfloor, \rfloor, \partial_H, a_H$ of ACP_r directly on \mathbb{F} . For $F, G \in \mathbb{F}$ we put

$$(i) \ F + G = \{[\epsilon, X] \mid [\epsilon, X] \in F \cap G\} \\ \cup \{\sigma \mid \sigma \in F \cup G\} \cup \{[\sigma, X] \mid \sigma \neq \epsilon \wedge [\sigma, X] \in F \cup G\}$$

In its first steps $F + G$ can refuse only those actions which can be refused by both F and G . In all subsequent steps $F + G$ behaves like $F \cup G$.

$$(ii) \ F \cdot G = \{[\sigma, X] \mid [\sigma, X] \in F\} \\ \cup \{\sigma_1 \sigma_2 \mid \sigma_1 \in F \wedge \sigma_2 \in G\} \\ \cup \{[\sigma_1 \sigma_2, X] \mid \sigma_1 \in F \wedge [\sigma_2, X] \in G\}$$

$F \cdot G$ first behaves like F and after successful termination of F in a trace σ_1 continues to behave like G .

$$(iii) \ F \parallel G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 \parallel \sigma_2\} \\ \cup \{[\sigma, X] \mid \exists [\sigma_1, Y] \in F, [\sigma_2, Y] \in G: \sigma \in \sigma_1 \parallel \sigma_2 \wedge X = Y - \{(a|b) \mid a, b \notin Y\}\}$$

where $\sigma_1 \parallel \sigma_2$ is the set of traces in A^+ defined inductively by:

$$a\sigma_1 \parallel b\sigma_2 = a \cdot (\sigma_1 \parallel b\sigma_2) \cup b \cdot (a\sigma_1 \parallel \sigma_2)$$

$$\cup [a|b] \cdot (\sigma_1 \parallel \sigma_2)$$

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

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

$$a \parallel b = \{ab, ba\} \cup [a|b]$$

with

$$[a|b] = \begin{cases} \{(a|b)\} & \text{if } a|b \neq \delta \\ \emptyset & \text{if } a|b = \delta \end{cases}$$

Thus $\sigma_1 \parallel \sigma_2$ is the set of successful traces resulting from merging and communicating with σ_1 and σ_2 . Note that $F \parallel G$ refuses an action c only if both F and G can refuse c and if it is impossible to obtain c as the result of a communication of two actions a and b which are not refused by F and G , respectively. This is the intuitive rationale behind the clause for $[\sigma, X]$ in the definition of $F \parallel G$. Clearly, $F \parallel G$ and $F|G$ are just variations of $F \parallel G$ differing only in their first actions.

$$(iv) F \parallel G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 \parallel \sigma_2\}$$

$$\cup \{[\epsilon, X] \mid [\epsilon, X] \in F\}$$

$$\cup \{[\sigma, X] \mid \sigma \neq \epsilon \wedge \exists [\sigma_1, Y] \in F, [\sigma_2, Y] \in G: \sigma \in \sigma_1 \parallel \sigma_2 \wedge X = Y - \{(a|b) \mid a, b \notin Y\}\}$$

where $\sigma_1 \parallel \sigma_2$ is the set of traces in A^+ defined inductively by:

$$a\sigma_1 \parallel \sigma_2 = a \cdot (\sigma_1 \parallel \sigma_2)$$

$$a \parallel \sigma_2 = \{a\sigma_2\}.$$

$$(v) F|G = \{\sigma \mid \exists \sigma_1 \in F, \sigma_2 \in G: \sigma \in \sigma_1 | \sigma_2\}$$

$$\cup \{[\epsilon, X] \mid \exists [\epsilon, Y_1] \in F, [\epsilon, Y_2] \in G: X \subseteq A - \{(a|b) \mid a \notin Y_1 \wedge b \notin Y_2\}\}$$

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

$$a\sigma_1 | b\sigma_2 = [a|b] \cdot (\sigma_1 \parallel \sigma_2)$$

$$a\sigma_1 | b = [a|b] \cdot \{\sigma_1\}$$

$$a | b\sigma_2 = [a|b] \cdot \{\sigma_2\}$$

$$a | b = [a|b].$$

$$(vi) \partial_H(F) = \{\sigma \mid \sigma \in F \text{ does not contain any } a \in H\}$$

$$\cup \{[\sigma, X \cup Y] \mid [\sigma, X] \in F, \sigma \text{ does not contain any } a \in H, \text{ and } Y \subseteq H\}.$$

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.

$$(vii) a_H(F) = \{a_H(\sigma) \mid \sigma \in F\}$$

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

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

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 .

The definitions of $+$ and \cdot above are as for \square and $;$ in BROOKES, HOARE & ROSCOE [9]. The definition of \parallel differs from that of [9] due to the different communication format in ACP_r . The operators $\sqcup, |, \partial_H, a_H$ are not present in [9].

5.2. The failure model.

The failure model of ACP_r is now given by the structure $\mathbb{F}(+, \cdot, \parallel, \sqcup, |, \partial_H, a_H)$.

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

$$\mathbb{H}_\delta(+, \cdot, \parallel, \sqcup, |, \partial_H, a_H) / \equiv_F \simeq \mathbb{F}(+, \cdot, \parallel, \sqcup, |, \partial_H, a_H)$$

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 $\llbracket g \rrbracket \in \mathbb{F}$ holds for every $g \in \mathbb{H}_\delta$. Also, by Definition 2.2.3.,

$$g \equiv_{\mathcal{F}} h \text{ iff } \mathcal{F}\llbracket g \rrbracket = \mathcal{F}\llbracket h \rrbracket$$

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, par abus de language, we denote also with \mathcal{F}). \mathcal{F} is surjective and behaves homomorphically over the operations $+, \cdot, \parallel, \sqcup, |, \partial_H$ 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

$$\mathbb{H}_\delta(\dots) \text{ to } \mathbb{F}(\dots). \square$$

6. ACP_r WITH 1-1 COMMUNICATION

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

6.1. 1-1 communication.

First we assume that A (with typical elements $a, b \in A$) is partitioned into $A = C \dot{\cup} I$ where C (with typical elements $c, d \in C$) is the set of *communicating* actions and I (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 $\parallel, \sqcup, |, \partial_H, a_H$ from x , using the axioms of ACP_r . (This is possible by virtue of an elimination theorem to this effect proved in [7] 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' :

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

(That $\alpha(x)$ is indeed well-defined in this way, follows from the confluency property of the rewriting procedure used in obtaining x' from x . This fact is for ACP also proved in [7] and is easily carried over to ACP_r .)

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

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

holds.

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

$$x = \sum_i c_i x_i + \sum_j d_j$$

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

Thus

$$x \parallel y = \sum_i c_i (x_i \parallel y) = \sum_j d_j y$$

which implies $\partial_C(x \parallel y) = \delta$. \square

For $|$ we now assume 1-1 communication, i.e. that there is a bijection $\phi: C \rightarrow C$ such that $c|\phi(c) \in I$ for every $c \in C$, and $a|b = \delta$ otherwise. Note that $c|\phi(c) \in I$ implies $c|\phi(c) \neq \delta$.

Next, we show that the definitions of parallel composition used in CCS and CSP are typical examples for 1-1 communication.

6.2. Milner's parallel composition $\parallel_{\mathcal{M}}$ in CCS [14]

Milner stipulates a bijection $\bar{\cdot}: C \rightarrow C$ on communicating actions 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 ' to denote the so-called *silent* action. (We will write here $\hat{\tau}$ not to confuse Milner's ' τ ' with the somewhat different rôle of τ in ACP_r [3,6].) In [14] the semantics of $\parallel_{\mathcal{M}}$ is defined operationally by means of transitions. Informally, $x \parallel_{\mathcal{M}} y$ is given by a nondeterministic interleaving of x and y , plus the communication of x and y via matching actions which then yield τ as a result.

It is easy to express this idea in ACP_r with 1-1 communication: take

$$I = \{\hat{\tau}\}, \quad \phi(c) = \bar{c} \text{ and } c|\bar{c} = \hat{\tau}.$$

Then we put

$$x \parallel_{\mathcal{M}} y =_{df} x|y.$$

Note that as soon as one component of $\parallel_{\mathcal{M}}$ terminates, the other component proceeds on its own. E.g.

$$c \parallel_{\mathcal{R}} \bar{c}d = \bar{c}cd + \bar{c}(cd + dc) + \hat{\tau}d.$$

Let in the following "CCS" stand for ACP , where \parallel realizes $\parallel_{\mathcal{R}}$ by taking I, ϕ and $|$ as described above.

6.3. Hoare's parallel composition $\parallel_{\mathcal{G}}$ in CSP [9]

In [9] Hoare proposes an operation $\parallel_{\mathcal{G}}$ modelling full synchronization with symmetric communication. We may introduce this operation to ACP , by the following set of axioms:

$(x + y) \parallel_{\mathcal{G}} z$	$= x \parallel_{\mathcal{G}} z + y \parallel_{\mathcal{G}} z$	FS1
$x \parallel_{\mathcal{G}} (y + z)$	$= x \parallel_{\mathcal{G}} y + x \parallel_{\mathcal{G}} z$	FS2
$ax \parallel_{\mathcal{G}} by$	$= \delta_{a,b} \cdot (x \parallel_{\mathcal{G}} y)$	FS3
$a \parallel_{\mathcal{G}} by$	$= \delta_{a,b} \cdot \delta$	FS4
$ax \parallel_{\mathcal{G}} b$	$= \delta_{a,b} \cdot \delta$	FS5
$a \parallel_{\mathcal{G}} b$	$= \delta_{a,b}$	FS6

TABLE 4

where $\delta_{a,b} = a$ if $a = b$ and $a \in C$, and $\delta_{a,b} = \delta$ otherwise.

6.3.1. REMARK. Axioms FS1 - FS6 specify a unique and totally defined binary operation $\parallel_{\mathcal{G}}$ on closed terms of ACP .

Note that by the axioms FS4 and FS5 the succesful termination of $x \parallel_{\mathcal{G}} y$ requires *joint succesful termination* of its components x, y . E.g.

$$cd \parallel_{\mathcal{G}} cd = cd, \text{ but } cd \parallel_{\mathcal{G}} cde = cd\delta.$$

We show that $\parallel_{\mathcal{G}}$ can be expressed in ACP , with 1-1 communication. Let $C = \{c_1, \dots, c_n\}$. Then we take

$$I = \{\hat{c} \mid c \in C\}, \phi(c) = c \text{ and } c \mid c = \hat{c}$$

where \hat{c} are new copies of the actions in C .

6.3.2. PROPOSITION. For closed terms x, y of ACP , with $\alpha(x), \alpha(y) \subseteq C$

$$x \parallel_{\mathcal{G}} y = C_I(\partial_C(x \parallel y))$$

holds where C_I abbreviates the composite renaming operator $C_I = c_{1\{\hat{c}_1\}} \circ \dots \circ c_{n\{\hat{c}_n\}}$.

PROOF. By Remark 6.3.1 it suffices to show that $C_I(\partial_C(x \parallel y))$ satisfies the axioms FS1 - FS6 for all closed terms with alphabet in C . This proof is simplified by the fact that

$$(*) C_I(\partial_C(x \parallel y)) = C_I(\partial_C(x \mid y))$$

holds due to Lemma 6.1.1.

Now it is clear that the distributivity of $+$ over \mid, ∂_C and C_I implies FS1 and FS2. Axiom FS3 (with $a, b \in C$) can be calculated as follows:

$$\begin{aligned} ax \parallel_{\mathcal{G}} by &= C_I(\partial_C(ax \parallel by)) = C_I(\partial_C(ax \mid by)) \\ &= C_I(\partial_C((a \mid b)(x \parallel y))) = \delta_{a,b} \cdot C_I(\partial_C(x \parallel y)) \\ &= \delta_{a,b} \cdot (x \parallel_{\mathcal{G}} y). \end{aligned}$$

FS4 - FS 6 are verified analogously. \square

Let in the following "CSP" stand for ACP , with the parallel composition $\parallel_{\mathcal{G}}$ (as axiomatized above) replacing \parallel , $\underline{\parallel}$ and $|$.

7. MAXIMAL TRACE RESPECTING CONGRUENCE

In Section 4 (Proposition 4.2.3) it was shown that failure equivalence $\equiv_{\mathcal{F}}$ is a congruence w.r.t. the operators of ACP . In this section we will prove that for ACP , with 1-1 communication failure equivalence is in fact the maximal trace respecting congruence. But first let us introduce the relevant concepts.

7.1. Preliminaries

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

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

$$x \equiv y \text{ implies } \mathcal{Q}[x] \equiv \mathcal{Q}[y]$$

for all terms $x, y \in \mathcal{T}$ and contexts $\mathcal{Q}[\xi] \in Ter(\Sigma)[\xi]$ with $\mathcal{Q}[x], \mathcal{Q}[y] \in \mathcal{T}$. 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 $\mathcal{Q}[\xi] \in Ter(\Sigma)[\xi]$ with $\mathcal{Q}[x], \mathcal{Q}[y] \in \mathcal{T}$ and $trace(\mathcal{Q}[x]) \neq trace(\mathcal{Q}[y])$.

7.1.1. PROPOSITION. For each $\mathcal{T} \subseteq Ter(\Sigma)$ 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} , $trace(\mathcal{Q}[x]) = trace(\mathcal{Q}[y])$ holds for every context $\mathcal{Q}[\xi] \in Ter(\Sigma)[\xi]$ with $\mathcal{Q}[x], \mathcal{Q}[y] \in \mathcal{T}$. But this contradicts the maximality of \equiv_2 .

Existence: Define \equiv , a binary relation on \mathcal{T} , as follows:

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

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

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. $\mathcal{R}[x] \neq \mathcal{R}[y]$ holds for $x, y \in \mathcal{T}_C$. If $trace(x) \neq trace(y)$, the trivial context

$$\mathcal{Q}[\xi] = \xi$$

will do. Suppose now that $trace(x) = trace(y)$ holds. Because of $x \not\equiv_{\mathcal{F}} y$ we can assume w.l.o.g. that there exists a failure pair $[\sigma, X]$ with

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

By the definition of \mathcal{F} , $[\sigma, X] \in \mathcal{R}[x]$ implies that there exists some ready pair $(\sigma, z) \in \mathcal{R}[x]$ with

$$X \subseteq \bar{z}$$

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 $[\sigma, C] \in \mathcal{R}[y]$ and therefore also $[\sigma, X] \in \mathcal{R}[x]$. 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} , $[\sigma, X] \notin \mathcal{R}[y]$ implies that for every such ready pair $(\sigma, Y) \in \mathcal{R}[y]$ there exists some

$$d \in X \cap Y.$$

Consider now a context of the form

$$\mathcal{Q}[\xi] = (c_{1(i_1)} \circ \dots \circ c_{n(i_n)} \circ \partial_C) (\xi \parallel \underbrace{\phi(\sigma) \cdot \sum_{d \in X \cap Y, (\sigma, Y) \in \mathcal{R}[y]} \phi(d)}_{\text{...}})$$

where we take $I = \{i_1, \dots, i_n\}$, $c_1, \dots, c_n \in C$, ϕ the bijection describing the 1-1 communication in ACP_r and $\phi(\sigma)$ the result of applying ϕ pointwise to σ . Note that $\mathcal{Q}[\xi]$ is uniquely determined by x and y except for the choice of the c_1, \dots, c_n in the renaming operators. Note that indeed $\mathcal{Q}[x], \mathcal{Q}[y] \in \mathcal{T}_C$ due to the presence of operators ∂_C and $c_{j(i_j)}$ in $\mathcal{Q}[\xi]$. We claim now that

$$(c_{1(i_1)} \circ \dots \circ c_{n(i_n)}) (\sigma \mid \phi(\sigma)) \cdot \delta \in trace(\mathcal{Q}[x]), \notin trace(\mathcal{Q}[y])$$

where $\sigma \mid \phi(\sigma)$ is understood by applying \mid pointwise to σ and $\phi(\sigma)$.

To prove this claim we first state a general observation about ready sets $\mathcal{R}[z]$ of closed terms z over ACP_r . Let $\sigma = a_1 \dots a_m$ and $Z = \{b_1, \dots, b_n\}$. Then

$$(\sigma, Z) \in \mathcal{R}[z]$$

iff there exist $x_1, \dots, x_m, y_1, \dots, y_n \in \text{Ter}(ACP_r)$ with

$$ACP_r \vdash x = a_1(a_2 \dots (a_m(b_1 y_1 + \dots + b_n y_n) + x_m) \dots + x_2) + x_1.$$

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 $\mathcal{Q}[\xi]$ by the communication operator $|$ which enforces synchronization.

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

$$(c_{1\{i_1\}} \circ \dots \circ c_{n\{i_n\}})(\sigma | \phi(\sigma)) \cdot \delta \in \text{trace}(\mathcal{Q}[x]).$$

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

Recalling from the proof that the trace separating contexts were either trivial, i.e.

$$\mathcal{Q}[\xi] = \xi,$$

or of the form

$$\mathcal{Q}[\xi] = (c_{1\{i_1\}} \circ \dots \circ c_{n\{i_n\}} \circ \partial_C)(\xi \parallel z)$$

for some closed term z over ACP_r and arbitrary $c_1, \dots, c_n \in C$ (with $I = \{i_1, \dots, i_n\}$), we can state the following:

7.2.2. COROLLARY. *The theorem (7.2.1) remains valid if we replace ACP_r by "CCS" or "CSP" in the sense of Section 6.2 resp. 6.3.*

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

7.2.3 REMARK. (Comparison with the work of de NICOLA & HENNESSY [10].)

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 somewhat related result in [10]. (Though the settings are quite different: here finite processes in ACP_r , there CCS with recursion, τ -steps and an additional constant Ω for the undefined state).

De NICOLA & HENNESSY [10] 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; the formal definitions, however, are somewhat more technical. Roughly, a test t is itself a process, and testing another process p essentially means running p and t in parallel (in the sense of CCS):

$$p \parallel_{\mathcal{R}} t. \quad (*)$$

The definition of passing a test involves a special action ω indicating success. In fact, three variants of *passing a test* are studied in [10], each leading to a different notion of equivalence on processes.

So individual processes p are tested in [10]. Using this terminology, our approach might be interpreted as testing *pairs* (p, q) of processes on trace equivalence. A test is here simply an arbitrary context $\mathcal{Q}[\xi]$. It is now very interesting that for the second equivalence \simeq_2 of [10] both ideas of testing seem to agree. According to [10] \simeq_2 coincides with the failure equivalence for the class of *strongly convergent* CCS terms. Roughly, the finite ACP_r processes with $\parallel_{\mathcal{R}}$ as parallel composition (cf. Section 6) are in this class. So \simeq_2 is at least very close with \equiv_g , the maximal trace respecting congruence. A hint why these two approaches might agree can be found in the proof of Theorem 7.2.1: there we see that it (essentially) suffices to consider restricted contexts of the form

$$\mathcal{Q}[\xi] = p \parallel z \quad (**)$$

which now resemble the test (*) of [10].

Summarizing, we find the notion of a maximal trace respecting congruence somewhat simpler than the definition of test in [10], though both ideas seem to agree for the restricted class of processes studied in this paper.

Finally, a comparison between the complete axiom system for \simeq_2 in [10] (leaving out the recursion part, the τ -part and the Ω -part) is not entirely straightforward, since [10] decomposes the equivalence \simeq_2 in a pre-order \sqsubseteq_2 and gives an axiomatisation for that pre-order. However, our axioms R1,2 and S (which form the part of *ACP*, specifically concerned with axiomatising failure equivalence) are immediate consequences of the proof of system of De NICOLA and HENNESSY in [10]:

(1) Axiom S in Table 3: $ax + a(y+z) = ax + a(y+z) + a(x+y)$ implies $ax + ay = ax + ay + a(x+y)$ by taking $z=y$; this is (D5) in [10]. 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 [10]. 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 [10] as follows.

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

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

$$bx + by + \tau(by+v) + \tau(bx+u) = \tau(bx+by+v) + \tau(bx+by+u)$$

$$bx + \tau(bx+u) = \tau(bx+u) \quad (D9)$$

$$by + \tau(by+v) = \tau(by+v) \quad (D9)$$

$$\tau(by+v) + \tau(bx+u) = \tau(bx+by+v) + \tau(bx+by+u)$$

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

$$a(by+v) + a(bx+u) = a(bx+by+v) + a(bx+by+u). \quad (N1)$$

Here N1,3 and D9 are axioms in [10].

(Axiom (R2): $a(b+u) + a(by+v) = a(b+by+v) + a(b+by+u)$ would be rendered in [10] as an instance of (R1):

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

8. PROCESSES WITH RECURSION AND ABSTRACTION: BISIMULATION VERSUS FAILURE EQUIVALENCE

8.1. Preliminaries.

In the preceding sections we have been exclusively concerned with the failure semantics for finite processes without 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 BROOKES, HOARE and ROSCOE [9], 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 process models obtained by bisimulation equivalence satisfy a useful abstraction principle: *Koomen's fair abstraction rule* (KFAR) as introduced in [4]. 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 [4] it was used to verify an alternating bit protocol), it is natural to ask whether KFAR is also compatible with the somewhat simpler failure semantics. More precisely, one can ask whether there exists a process model which for finite processes agrees with the failure semantics and for infinite processes satisfies KFAR. Rather surprisingly 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, + 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, which specify the properties of $+$ and \cdot . T1 is the simplest of Milner's τ -laws [14] (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
$\tau_1(\tau) = \tau$	TI1
$\tau_1(a) = a$ if $a \notin I$	TI2
$\tau_1(a) = \tau$ if $a \in I$	TI3
$\tau_1(x + y) = \tau_1(x) + \tau_1(y)$	TI4
$\tau_1(xy) = \tau_1(x)\tau_1(y)$	TI5

TABLE 5. BPA + T1 + TI1-5.

In addition, Table 5 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. [5] and [6]).

R1 is the axiom for the readiness semantics (see Table 3):

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

The *recursive specification principle* RSP states that guarded systems E of recursive equations have unique solutions (see [4] or [1]):

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

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

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

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:

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

The premiss of $KFAR_1$ says that x has an infinite i -trace; see Figure 18. 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 .

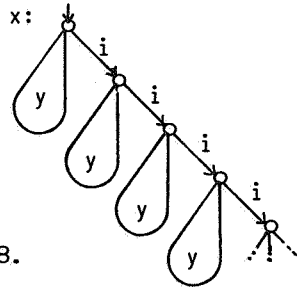


FIGURE 18.

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 20. These two graphs are in fact identical.

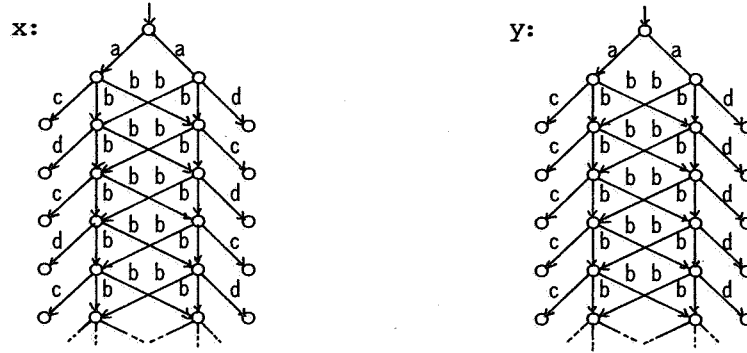


FIGURE 20.

Formally: PROOF OF THE CLAIM. Consider the system E_3 of guarded recursion equations:

$$E_3 \begin{cases} z = az_1 + az_2 \\ z_1 = c + bz_1 + bz_2 \\ z_2 = d + bz_1 + bz_2. \end{cases}$$

(This system corresponds with the graph in Figure 20.) Now

$$\begin{aligned} x &= ax_1 + ax_2 = a(c + bx_2) + a(d + bx_1) = (\text{by R1}) \\ &a(c + bx_2 + bx_1) + a(d + bx_1 + bx_2) = az'_1 + az'_2 \end{aligned}$$

where

$$z'_1 = c + bx_2 + bx_1 \text{ and } z'_2 = d + bx_1 + bx_2.$$

Further,

$$\begin{aligned} z'_1 &= c + bx_2 + bx_1 = c + b(d + bx_1) + b(c + bx_2) = \\ &(\text{by R1}) c + b(bx_1 + bx_2 + c) + b(bx_1 + bx_2 + d) = \\ &c + bz'_1 + bz'_2, \end{aligned}$$

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 $z''_1 = c + by_1 + by_2$ and

$z_2'' = d + by_1 + by_2$ 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 21.

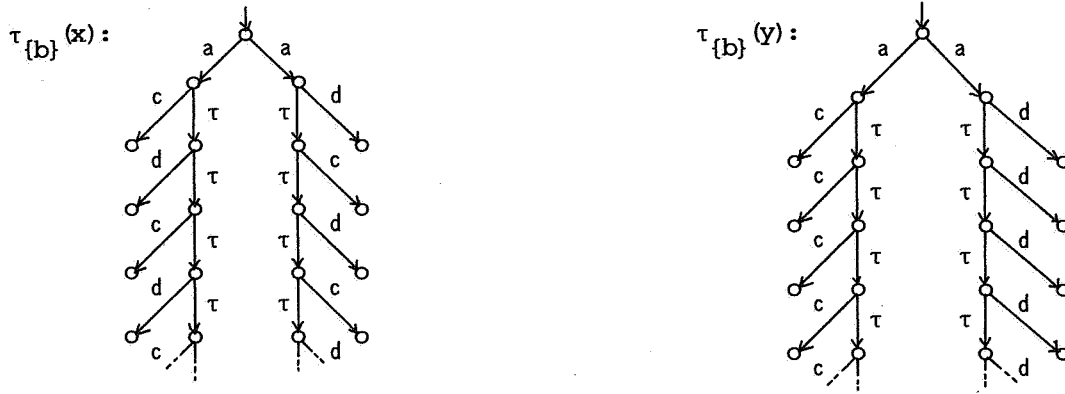


FIGURE 21.

Next we apply KFAR on $\tau_{\{b\}}(x)$ and $\tau_{\{b\}}(y)$ and obtain $a(c+d)$ resp. $ac+ad$. This can be seen graphically: KFAR shrinks the infinite τ -traces to a point, obtaining the graphs as in Figure 22.

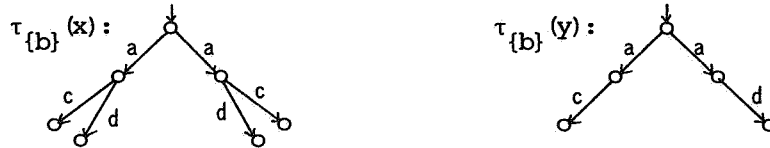


FIGURE 22.

Formally:

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

Further, $x_1 = bx_2 + c$, $x_2 = bx_1 + d$ yields by $KFAR_2$:

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

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

Hence from (*):

$$\tau_{\{b\}}(x) = a\tau(c+d) + a\tau(c+d) = (\text{by } T1 \text{ in Table 5})$$

$$a(c+d) + a(c+d) = a(c+d).$$

Next consider y :

$$\tau_{\{b\}}(y) = a.\tau_{\{b\}}(y_1) + a.\tau_{\{b\}}(y_2) \quad (**)$$

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.

REFERENCES

- [1] BAETEN, J.C.M., J.A. BERGSTRÄ & J.W. KLOP, *On the consistency of Koomen's Fair Abstraction Rule*, Report CS-R8511, Centrum voor Wiskunde en Informatica, Amsterdam 1985.
- [2] BERGSTRÄ, J.A. & J.W. KLOP, *An abstraction mechanism for process algebras*, Report IW 231/83, Mathematisch Centrum, Amsterdam 1983.
- [3] BERGSTRÄ, J.A. & J.W. KLOP, *Algebra of communicating processes with abstraction*, Theoret. Comput. Sci. 37 (1985) 77-121.
- [4] BERGSTRÄ, J.A. & J.W. KLOP, *Verification of an alternating bit protocol by means of process algebra*, Report CS-R8404, Centrum voor Wiskunde en Informatica, Amsterdam 1984.
- [5] BERGSTRÄ, J.A. & J.W. KLOP, *A complete inference system for regular processes with silent moves*, Report CS-R8420, Centrum voor Wiskunde en Informatica Amsterdam 1984.
- [6] BERGSTRÄ, J.A. & J.W. KLOP, *Algebra of Communicating Processes*, To be published in: Proceedings of the CWI Symposium Mathematics and Computer Science (eds. J.W. de Bakker, M. Hazewinkel and J.K. Lenstra), North-Holland, Amsterdam 1985.
- [7] BERGSTRÄ, J.A. & J.W. KLOP, *Process algebra for synchronous communication*, Information and Control, Vol. 60, Nos. 1-3, p. 109-137, 1984.
- [8] 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, 83-96, Springer-Verlag, New York/Berlin, 1983.
- [9] BROOKES, S., C. HOARE & W. ROSCOE, *A Theory of Communicating Sequential Processes*, J. Assoc. Comput. Mach. 31, No. 3, p. 560-599, 1984.
- [10] DE NICOLA, R. & M.C.B. HENNESSY, *Testing equivalences for processes*, TCS Vol. 34, Nrs. 1, 2, p. 83-133, 1984.
- [11] HENNESSY, M., *Synchronous and Asynchronous Experiments on Processes*, Report CSR-125-82, Univ. of Edinburgh, 1982.
- [12] HOARE, C.A.R., *Communicating sequential processes*, Comm. ACM 21 666-677, 1978.
- [13] HOARE, C.A.R., *A model for communicating sequential processes*, in: On the Construction of Programs (R.M. McKeag and A.M. McNaghton, Eds.), pp. 229-243, Cambridge Univ. Press, London/New York, 1980.
- [14] MILNER, R., *A Calculus of Communicating Systems*, Lecture Notes in Computer Science No. 92, Springer Verlag, New York/Berlin, 1980.
- [15] MILNER, R., *Calculi for synchrony and asynchrony*, Theoret. Comput. Sci. 25 (1983), p. 267-310.
- [16] OLDEROG, E.R. & C.A.R. HOARE, *Specification-oriented semantics for communicating processes*, in Proc. 10th Int. Colloq. Automat. Lang. & Programming, Barcelona, 561-572, Lecture Notes in Computer Science No. 154, Springer Verlag New York/Berlin, 1983; expanded version, Technical Monograph PRG-37, Oxford Univ. Comput. Lab., February 1984.
- [17] PARK, D.M.R., *Concurrency and automata on infinite sequences*, in: Proc. 5th GI (Gesellschaft für Informatik) Conference, Springer LNCS 104, 1981.
- [18] ROUNDS, W.C. & S.D. BROOKES, *Possible futures, acceptances, refusals, and communicating processes*, in: Proceedings of 22nd IEEE Symposium on Foundations of Computer Science, Nashville, Tennessee (IEEE Computer Society Press, 1981) p. 140-149.
- [19] WINSKEL, G., *Synchronisation trees*, in: Proc. 10th ICALP, Barcelona (ed. J. Díaz), Springer LNCS 154, p. 695-711, 1983.