**Centrum voor Wiskunde en Informatica**
Centre for Mathematics and Computer Science

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

# Failures without chaos: A new process semantics for fair abstraction

Computer Science/Department of Software Technology Report CS-R8625 August

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.).

6gF11, 6gF2, 6gF32, 6gF43

# Failures without Chaos:

# A New Process Semantics for Fair Abstraction

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

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

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

*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, failure semantics, bisimulation semantics.
*Note:* (i)The research of J.A. Bergstra and J.W. Klop is partially supported by ESPRIT project 432: Meteor.
(ii) This report will be published in the Proceedings of the Working Conference on the Formal Description of Programming Concepts, August 1986, Gl. Avernaes, Denmark (ed. M. Wirsing), North-Holland.
The present report is partly a condensed version of the technical report from the same authors 'Failure Semantics with Fair Abstraction', Report CS-R8609, Centre for Mathematics and Computer Science, Amsterdam; also some new material is included.

ABSTRACT
We propose a new process semantics that combines the advantages of fair abstraction from internal process activity with the simplicity of failure semantics. The new semantics is obtained by changing the way the original failure semantics of Brookes, Hoare and Roscoe or the equivalent acceptance semantics of de Nicola and Hennessy deal with infinite internal process activity, known as divergence. We work in an algebraic setting and develop the new semantics stepwise, thereby systematically comparing previous proposals.

Report CS-R8625
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

## 1. Introduction.

The concept of abstraction is most important for mastering the complexity of process verification. The reason is that abstraction allows larger processes to be constructed and verified hierarchically as systems of smaller ones. For example, imagine a system SYS consisting of three components connected as in Figure 1(a). Typically, the task of verification is then to prove that such a system behaves like a much simpler process SPEC serving as system specification.

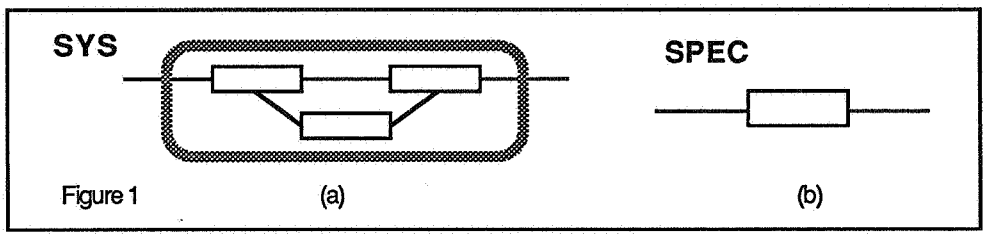



Figure 1

This proof is possible only by "abstracting" from the internal structure of the system as shown within the confined area in Figure 1(a) and comparing the specification with the remaining external behaviour of the system outside the confined area.

Combining ideas of Milner [Mi 1] and Hoare [Ho], abstraction can be modelled by distinguishing two kinds of actions in a process, viz. *external* or *observable* actions, and *internal* or *hidden* actions, and by introducing an explicit hiding operator that transforms observable actions into internal ones.

One of the difficult issues linked with abstraction is how to deal with divergence, i.e. with the capability of a process to execute an infinite sequence of internal actions. That is what our paper is about. We work in an algebraic setting where the intended semantics of processes is described by algebraic laws or axioms, and where process models prove the consistency of these axiomatisations (cf. [BK 1-4]).

We are specifically interested in the notion of fair abstraction as exemplified in Koomen's Fair Abstraction Rule (KFAR) [Ko, BBK 2]. This rule has proved particularly useful in algebraic protocol verification because it can deal with unreliable, but fair transmission media. For example, in [BK 3] and [Va] Alternating Bit and Sliding Window Protocols were verified using Koomen's Fair Abstraction Rule. Formally, KFAR allows to condense a divergence into a single internal action. KFAR is justified in Milner and Park's bisimulation semantics [Mi 1,3, Pa]. This is a very discriminating semantics where (divergence free) processes can be identified only if their global branching structure coincides [BBK 2].

Starting from the idea that only linear histories or traces of communications with a process can be observed, bisimulation semantics is too discriminating, and the linear failure semantics of [BHR] or the equivalent ("must"-version of) acceptance semantics in [dNH, He] seems appropriate (cf.[Pn]). This brings us to the main question of our paper:

*Can the advantages of fair abstraction be combined with the simplicity of a trace consistent, linear process semantics?*

At first sight, the answer seems to be "no". Firstly, the original failure semantics of [BHR] equates divergence with the "catastrophic" process CHAOS that makes any distinction of the subsequent process behaviour impossible. Moreover, in [BKO] it was proved that failure semantics is inconsistent with the rule KFAR in the sense that it forces us to identify finite processes which are distinguished in failure semantics.

Nevertheless, in this paper we shall present a new failure semantics without CHAOS which admits a restricted rule KFAR$^-$ for the fair abstraction of so-called unstable divergence. It is interesting to note that KFAR$^-$ turns out to be sufficient for the protocol verifications in [BK 3, Va]. We demonstrate the usefulness of KFAR$^-$ by treating a small, idealised protocol due to [Par]. The proposed semantics differs also from all versions of acceptance semantics discussed in [dNH, He].

In fact, we shall present a systematic analysis of axiom systems and semantic models centering around the notions of abstraction and divergence. Both the exposition and the elegance of the various axiomatisations are greatly enhanced by introducing a process $\Delta$ (pronounced "delay") modelling divergence. $\Delta$ is inspired by Milner's delay operator in SCCS [Mi 3]. With the symbols $\delta$ and $\tau$ denoting deadlock and internal action, the main theories about divergence can be characterised by simple equations about $\Delta$:

- as starting point a (new) bisimulation semantics with explicit divergence $\Delta$,

- bisimulation semantics of [Mi 1, Pa] with fair abstraction: $\Delta = \tau$,

- a (new) failure semantics with explicit divergence $\Delta$,

- failure semantics of [BHR, BR] with catastrophic divergence which is equivalent to the "must"-version of acceptance semantics in [dNH, He]: $\Delta\delta = \Delta$,

- finally, the main new failure semantics with fair abstraction of unstable divergence: $\Delta\tau = \tau$.

The details are explained in the rest of this paper.

## 2. Bisimulation Semantics with Explicit Divergence: $BS_\Delta$

### 2.1. Algebraic setting.

The signature of processes contains a set A of *atomic processes* a,b,c,... modelling observable atomic actions, and the following *characteristic processes*:

- $\delta$ - modelling *deadlock*,
- $\varepsilon$ - empty process modelling *termination*,
- $\tau$ - modelling an *internal* or *hidden* action.

As process operations we admit, for $I \subseteq A$:

- $+$ - binary infix operation modelling *nondeterminism*,
- $.$ - binary infix operation modelling *sequential composition*,
- $\tau_I$ - unary infix operation modeling *abstraction* from or *hiding* of all actions in I.

The above notation is chosen for its conciseness. Deadlock $\delta$ corresponds to NIL in CCS [Mi 1] and STOP in TCSP [BHR, Ho]. Termination $\varepsilon$ and sequential composition x.y are not present in CCS but in TCSP where they are denoted by SKIP and x;y, respectively. Often we simply write xy instead of x.y. The symbols $\tau$ and $+$ are taken from CCS. For finite sums we use the convenient abbreviation $\Sigma$, e.g.

$$\Sigma_{k \in \{1,...,n\}} x_k = x_1 + ... + x_n.$$

(This notation is justified because $+$ is commutative and associative: see the axioms in Table 1.)

The hiding operator $\tau_I(x)$ is from TCSP where it is written as $x\backslash I$. The notation $\tau_I(x)$ is chosen to remind us that in x all actions $a \in I$ are renamed into $\tau$ (see Table 1 again). In CCS hiding is always coupled with parallel composition, a solution which would obscure our present analysis. In fact, we do not consider parallel composition $\|$ here because the problems with abstraction and divergence arise already without this operation. However, we see no difficulties in adding the $\|$ of [BK 1] to the results in our paper.

Observably infinite processes are introduced by recursive definitions. We consider possibly infinite, guarded systems

$$E(x) = \{x_k = T_k(x) \mid k \in K\}$$

of recursive equations where x is a finite or infinite set $x = \{x_k \mid k \in K\}$ of variables with index set K and where each right hand side term $T_k(x)$ is constructed from variables of x, constants of $A \cup \{\delta, \varepsilon, \tau\}$ and the operations $+$ and $.$ but not $\tau_I$. *Guarded* means that for every $k \in K$ there is some expansion $S_k(x)$ of $x_k$ by E(x) such that each occurrence of $x_k$ in $S_k(x)$ is preceded by some

observable action $a \in A$. Thus a hidden action $\tau$ is not sufficient for guardedness. For example, the system $\{ x_1 = x_2, x_2 = a.x_1\}$ is guarded, but $\{x_1 = \tau.x_1\}$ is not.

We require that such systems have solutions (Recursive Definition Principle RDP) which are moreover unique (Recursive Specification Principle RSP). These principles together with the basic axioms about $\varepsilon$, $\delta$, $\tau$ and $+, ., \tau_I$ are given in Table 1 (next page).

Axioms A1-5 describe the general properties of . and +. Axioms A6 and A7 deal with deadlock $\delta$: by A7 there is no subsequent behaviour possible after $\delta$, and by A6 deadlock is discarded in the presence of another nondeterministic alternative. A6 is typical for global nondeterminism (cf. [BMOZ]). Axiom A8 characterises $\varepsilon$ as the purely terminating process. Abstraction is described in two groups of axioms: TI1-4 describes the hiding operator $\tau_I$ as a simple renaming operator that renames every action $a \in I$ into $\tau$; how to deal with $\tau$ is then described by Milner's $\tau$-laws T1-3 [Mi 1].

To illustrate the use of these axioms we derive the following consequence.

PROPOSITION 1. *For any* $z$ *the term* $x = \tau(z + y)$ *solves the (unguarded) equation* $x = \tau x + y$.

PROOF. $x =_{[\text{def. } x]} \tau(z + y) =_{[T2]} \tau(z + y) + z + y =_{[A3]} \tau(z + y) + z + y + y =_{[T2]}$

$\tau(z + y) + y =_{[T1]} \tau\tau(z + y) + y =_{[\text{def. } x]} \tau x + y.$ □

Proposition 1 implies that in any trace consistent model (see section 2.4) of the axioms in Table 1 the equation $x = \tau x + y$ has infinitely many solutions. This explains why we restrict ourselves to guarded systems of equations when specifying processes via RDP and RSP.

**2.2. Abstraction and Divergence.**

The main concern of our paper is how to apply the abstraction operator $\tau_I$ to recursively defined, infinite processes. Consider the equation

$$x = i.x + y \qquad (*)$$

with $i \in I$. In fact, take $I = \{i\}$. Applying $\tau_{\{i\}}$ to x with the help of TI1-4 just gives

$$\tau_{\{i\}}(x) = \tau.\, \tau_{\{i\}}(x) + \tau_{\{i\}}(y),$$

Table 1

*General Properties:*

| | |
|---|---|
| $x + y = y + x$ | A1 |
| $(x + y) + z = x + (y + z)$ | A2 |
| $x + x = x$ | A3 |
| $(x + y)z = xz + yz$ | A4 |
| $(xy)z = x(yz)$ | A5 |
| $\delta + x = x$ | A6 |
| $\delta x = \delta$ | A7 |
| $\varepsilon x = x\varepsilon = x$ | A8 |

*Abstraction:*

| | |
|---|---|
| $\tau_I(a) = \tau$ where $a \in I$ | TI1 |
| $\tau_I(\gamma) = \gamma$ where $\gamma \notin I$ | TI2 |
| $\tau_I(x + y) = \tau_I(x) + \tau_I(y)$ | TI3 |
| $\tau_I(x.y) = \tau_I(x).\tau_I(y)$ | TI4 |
| | |
| $\alpha\tau = \alpha$ | T1 |
| $\tau x + x = \tau x$ | T2 |
| $\alpha(\tau x + y) = \alpha(\tau x + y) + \alpha x$ | T3 |

*Recursion:*

| | |
|---|---|
| $\exists x: E(x)$ | RDP |
| $\underline{E(x) = E(y)}$ <br> $x = y$ | RSP |

In the above axioms, a ranges over A, $\alpha$ over $A \cup \{\tau\}$, $\gamma$ over $A \cup \{\delta, \varepsilon, \tau\}$ and E(x) stands for a guarded system of equations.

an unguarded recursive equation which by Proposition 1 has infinitely many solutions. But looking at the unique process x satisfying (*) the behaviour of $\tau_{\{i\}}(x)$ should be clear intuitively: $\tau_{\{i\}}(x)$ either takes (after finitely many i-steps) the y-branch to behave like $\tau_{\{i\}}(y)$ or it pursues an infinite sequence of hidden i-steps, i.e. it diverges.

Thus abstraction (or hiding) and divergence are intimately linked with each other. To express this fact, we use an idea of Milner [Mi 3] and introduce a new characteristic process:

$\Delta$ - pronounced "delay" modelling divergence.

The application of $\tau_I$ to recursive processes can now be explained by the following Delay Rule DE, which is parameterised with $n \geq 1$:

$$\frac{\forall k \in \mathbb{Z}_n: \quad x_k = i_k.x_{k+1} + y_k, \quad i_k \in I}{\tau_I(x_0) = \Delta.\tau_I(\Sigma_{k\in\mathbb{Z}_n}\ y_k)} \qquad DE_n$$

Here $\mathbb{Z}_n = \{0,...,n-1\}$ and addition in the subscripts works modulo n. For n = 1 the rule applies to (*), yielding

$$\tau_{\{i\}}(x) = \Delta.\tau_{\{i\}}(y).$$

Putting $y = \varepsilon$ we obtain $\Delta = \tau_{\{i\}}(x)$ which by the previous axioms yields:

COROLLARY 1. (i) $\Delta = \Delta + \varepsilon$, (ii) $\Delta = \tau\Delta$, (iii) $\Delta = \tau\Delta + \varepsilon$.

PROOF. First we derive (iii):

$$\Delta = \tau_{\{i\}}(x) =_{[\text{def. x}]} \tau_{\{i\}}(ix + \varepsilon) =_{[TI1\text{-}4]} \tau.\tau_{\{i\}}(x) + \varepsilon = \tau\Delta + \varepsilon.$$

Now (i) and (ii) follow immediately:

$$\Delta = \tau\Delta + \varepsilon =_{[A3]} \tau\Delta + \varepsilon + \varepsilon = \Delta + \varepsilon,$$
$$\Delta = \tau\Delta + \varepsilon =_{[A3]} \tau\Delta + \tau\Delta + \varepsilon = \tau\Delta + \Delta =_{[T2]} = \tau\Delta.$$

□

Equation (i) says $\Delta$ may terminate, (ii) says $\Delta$ can perform arbitrarily many hidden $\tau$-steps, and (iii) combines both properties. We add one further axiom about $\Delta$ saying that $\Delta$ has no observable actions:

$$\tau_I(\Delta) = \Delta \qquad TI5$$

The resulting axiom system, i.e. Table 1 + $DE_n$ + TI5, we call $BS_\Delta$.

### 2.3. A model for $BS_\Delta$.

Axiomatic systems must be *logically consistent*, i.e. possess a model satisfying their axioms. Well-known are tree models for processes [Mi 1, He]. Here we build a model by dividing out a suitable equivalence on the domain of process graphs which represent the state-transition diagrams of nondeterministic automata.

A *process graph* is a rooted, directed multigraph. In this paper a process graph will always be finitely branching. Its nodes (states) are marked "open" (final states) or "closed" (other states) and its edges (transitions) are labelled by elements of $A \cup \{\tau\}$. Process graphs may contain cycles but not at their roots.

Let $\mathcal{G}$ be the set of all finitely branching process graphs, with $g,h \in \mathcal{G}$. Atomic and characteristic processes are represented as follows:

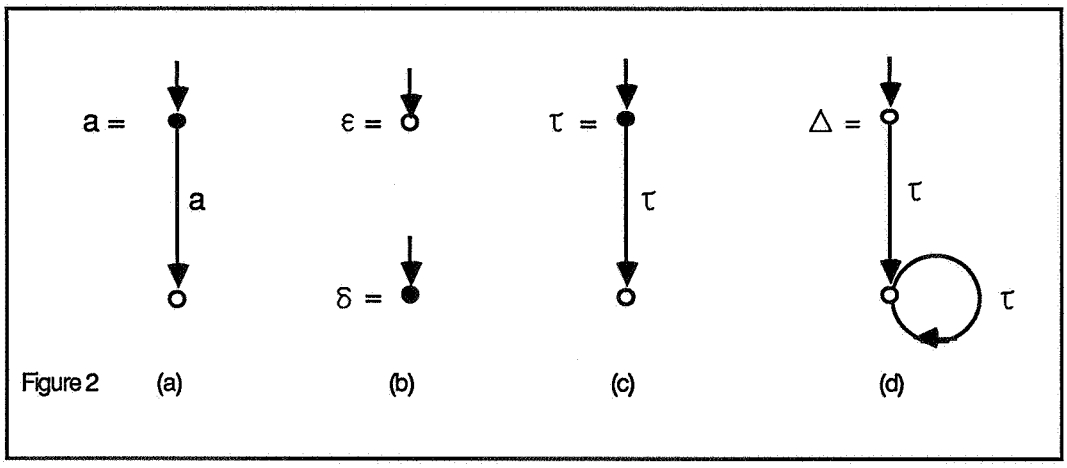



Figure 2 (a) (b) (c) (d)

The sum g + h is obtained by identifying the roots of g and h where "open" wins, the product g.h by appending h at all "open" nodes of g, the abstraction $\tau_I(g)$ by changing all labels $a \in I$ in g to $\tau$.

As equivalence on $\mathcal{G}$ we take a version of Milner's original observational equivalence [Mi 1] or better Park's bisimilarity [Pa, Mi 3] which keeps track of all divergences, called here *Δ-bisimilarity* $\leftrightarrows_\Delta$.

First we have to distinguish between different types of nodes in process graphs. A node is a *divergence node* if it is the starting point of an infinite path with all edges labelled by $\tau$. A node is called *exit node* if it is the starting point of a finite path with all edges labelled by $\tau$ and ending in an open node (modelling a final state). As usual we write for nodes s,t of a process graph g and a trace $\sigma \in A^*$:

$$s \Longrightarrow^{\sigma}_{g} t$$

if there exists a finite path from s to t in g labelled by a sequence $\lambda_1,...,\lambda_n \in (A \cup \{\tau\})^*$ such that $\sigma$ results from $\lambda_1,...,\lambda_n$ by skipping all $\tau$'s.

For example, consider the following process graph with nodes r, s, t, u:

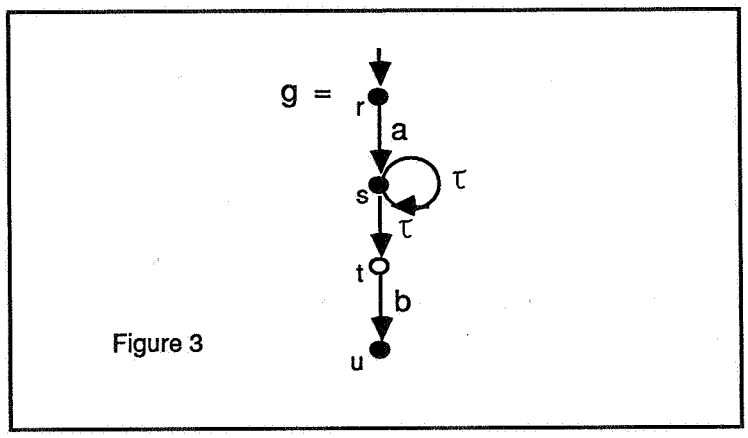



Figure 3

Then s and t are exit nodes, but not r and u. Moreover, g contains exactly one divergence node, viz. s. Finally, we have

$$r \Longrightarrow^{a}_{g} s,\ r \Longrightarrow^{a}_{g} t,\ r \Longrightarrow^{ab}_{g} u.$$

A *weak bisimulation* [Pa, Mi 3] between process graphs g and h is a relation R between the nodes of g and h satisfying the following conditions:

(B1) The roots of g and h are related.
(B2) Whenever sRt and $s \Longrightarrow^{\sigma}_{g} s'$ then, for some node t' in h, s'Rt' and $t \Longrightarrow^{\sigma}_{h} t'$.
(B3) Conversely, whenever sRt and $t \Longrightarrow^{\sigma}_{h} t'$ then, for some node s' in g, s'Rt' and $s \Longrightarrow^{\sigma}_{g} s'$.

Now, two process graphs g and h are *Δ-bisimilar*, abbreviated $g \leftrightarrows_{\Delta} h$, if there exists a weak bisimulation R additionally satisfying the following conditions:

(B4) A root may only be related to a root.
(B5) An exit node may only be related to an exit node.
(B6) A divergence node may only be related to a divergence node.

Also Milner considers a version of bisimilarity which takes divergence into account ([Mi 2]). However, his version differs from ours. For example in Milner's context the equation $\Delta(ab + a\Delta) = \Delta ab$ holds, but not with our definition of bisimilarity. Condition B5 is not needed in Milner's framework of CCS ([Mi 1]) because the notion of successful termination is not considered there. However, recall that Milner's definitions of observational equivalence do not yield a congruence w.r.t. the nondeterminism operator +. By adding condition B4, we avoid this problem.

We state without proof (see [BBK 2] for a similar proof):

THEOREM 1. (i) $\leftrightarrows_\Delta$ *is a congruence w.r.t. the operations in* $\mathcal{G}$.
(ii) $\mathcal{G}/\leftrightarrows_\Delta \models BS_\Delta$, *i.e. modulo* $\Delta$*-bisimilarity the process graphs satisfy all axioms of* $BS_\Delta$.

Notation: $\mathcal{A}(BS_\Delta) = \mathcal{G}/\leftrightarrows_\Delta$. It is well-known that bisimilarity preserves the branching structure of process graphs.

**2.4. Trace Consistency.**
Exhibiting a model for a process axiomatisation rules out logical contradictions, but it does not guarantee that the axiomatisation captures the operational intuitions about processes. The simplest such intuition is that of a trace. We therefore require that a process axiomatisation T is *trace consistent*, i.e. whenever

$$T \vdash x = y$$

holds for two finite and closed process terms x and y involving only $A \cup \{\varepsilon, \delta, \tau\}$ and + and . then their set of complete traces must agree:

$$\underline{\text{trace}}(x) = \underline{\text{trace}}(y).$$

A *complete trace* is a trace ending with a symbol $\surd$ or $\delta$ indicating successful termination ($\sigma\surd$, $\sigma \in A^*$) or deadlock ($\sigma\delta$, $\sigma \in A^*$). Formally, the set $\underline{\text{trace}}(x)$ is defined as follows. First, *normalise* the finite, closed term x by applying the "rewrite rules" of Table 2. (In fact, we work with finite closed terms x modulo A1, A2, A5 of Table 1.)

Table 2

| | | |
|---|---|---|
| $(x + y)z$ | $\rightarrow$ | $xz + yz$ |
| $\delta + x$ | $\rightarrow$ | $x$ |
| $\delta x$ | $\rightarrow$ | $\delta$ |
| $\varepsilon x$ | $\rightarrow$ | $x$ |
| $x\varepsilon$ | $\rightarrow$ | $x$ |
| $\alpha\tau$ | $\rightarrow$ | $\alpha$ |

Above $\alpha$ ranges over $A \cup \{\tau\}$

For normalised x, $\underline{\text{trace}}(x)$ is defined inductively as shown in Table 3.

Table 3

$\underline{trace}(\delta) = \{\delta\}$
$\underline{trace}(\varepsilon) = \{\surd\}$
$\underline{trace}(\tau) = \{\surd\}$
$\underline{trace}(a) = \{a\surd\}$
$\underline{trace}(y + z) = \underline{trace}(y) \cup \underline{trace}(z)$
$\underline{trace}(a.y) = a.\underline{trace}(y)$
$\underline{trace}(\tau.y) = \underline{trace}(y)$

Above a ranges over A

All our models will be such that they imply trace consistency. Thus:

PROPOSITION 2. $BS_\Delta$ *is trace consistent.*

## 3. Bisimulation Semantics with Fair Abstraction: BS.

### 3.1. Fair Abstraction.

The Delay Rule $DE_n$ allows us to apply abstraction to infinite recursive processes, but the result always contains $\Delta$ signalling divergence. This is disturbing because often one can assume *fairness* in the sense that a process will never stay forever in a cycle of hidden $\tau$-steps but will eventually exit it. Can we axiomatise this assumption? Yes, simply by adding the axiom

$$\Delta = \tau$$

to the system $BS_\Delta$. We will examine the resulting system BS. The Delay Rule $DE_n$ now specialises to Koomen's Fair Abstraction Rule $KFAR_n$

$$\frac{\forall k \in \mathbb{Z}_n: \quad x_k = i_k.x_{k+1} + y_k, \quad i_k \in I}{\tau_I(x_0) = \tau.\tau_I(\Sigma_{k\in\mathbb{Z}_n}\ y_k)} \qquad KFAR_n$$

of [BBK 2]. KFAR formalises an observation by Milner about his calculus CCS [Mi 1] and was first used by C.J. Koomen of Philips Research, Eindhoven, in a formula manipulation system for CCS (see also [Ko]). *Fair abstraction* means that $\tau_I(x_0)$ will eventually exit the hidden $i_0$-$i_1$-...-$i_{n-1}$ cycle.

A model for BS is obtained from the process graphs $\mathcal{G}$ by dividing out a slight variation $\triangleq$ of the original bisimilarity relation of [Pa, Mi 3]: $\mathcal{A}(BS) = \mathcal{G}/\triangleq$ (see [BBK 2]). In fact, $\triangleq$ is

defined as $\triangleq_\Delta$, but without condition B6. Thus BS is essentially the semantics of [Mi 1,3, Pa].

### 3.2. Protocol Verification in BS.

The rule KFAR has proved a crucial tool in a number of algebraic protocol verifications including Alternating Bit [BK 3] and Sliding Window Protocols [Va]. This is because KFAR can capture well the idea of a faulty, but fair transmission medium. We will demonstrate this by treating a small example.

Intuitively, a protocol is a set of rules describing how two processes, a sender and a receiver, communicate with each other over a transmission medium [Ta]. The task of protocol verification is to show that sender and receiver achieve a reliable communication despite a possibly unreliable medium that may lose or corrupt the messages sent [vB, Ha]. Formally, a protocol can be described as a system constructed hierarchically from the sender process, the receiver process, and a process modelling the medium [BK 2, Par]. The rules of communication are then incorporated in these processes.

We consider now an idealised protocol P essentially due to [Par]. Its components, the sender S, the medium M and the receiver R, are connected via directed communication channels named send, in, error, out, ack and rec:

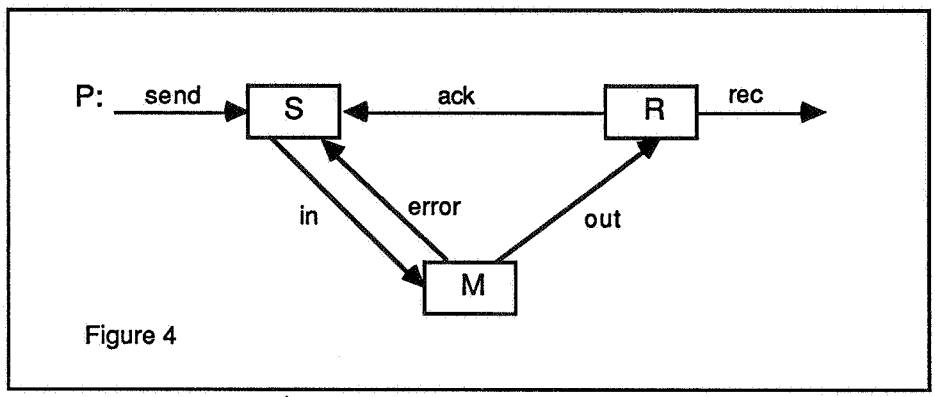



Figure 4

Their behaviour is described by the following recursive equations:

$$S = send?.S_0$$
$$S_0 = in!.(ack?.S + error?.S_0)$$
$$M = in?.(out!.M + error!.M)$$
$$R = out?.rec!.ack!.R$$

For illustration, we exhibit in Figure 5 (next page) the process graphs denoted by S, M and R.

Using the CSP notation [Ho], input of a message along a communication channel ch $\in$ {send,...,rec} is denoted by the action ch? and output by ch!. Intuitively, the sender S inputs a message from channel send and forwards it to the medium along channel in. If S gets an acknowledgement ack (from the receiver), it can input the next message from send. If, however, S gets the indication error representing loss or corruption of the message (inside the medium), S

retransmits the present message before inputting a new one from channel send. The intuition about M and R can be explained in a similar way.

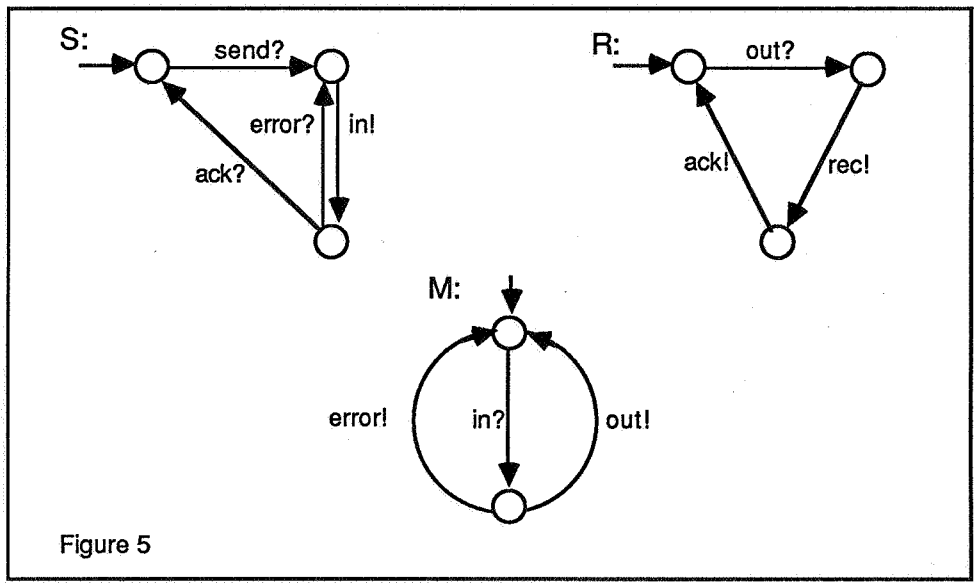



Figure 5

The protocol P explains how the components S, M, R work together. As in [BK 3] P can be described using parallel composition || and encapsulation $\partial_H$:

$$P = \partial_H ( S \,||\, M \,||\, R ).$$

We omit these details here and record only the result of this construction:

$$P = send?.P_0$$
$$P_0 = \underline{in}.P_1$$
$$P_1 = \underline{error}.P_0 + \underline{out}.rec!.\underline{ack}.P$$

denoting the process graph in Figure 6.

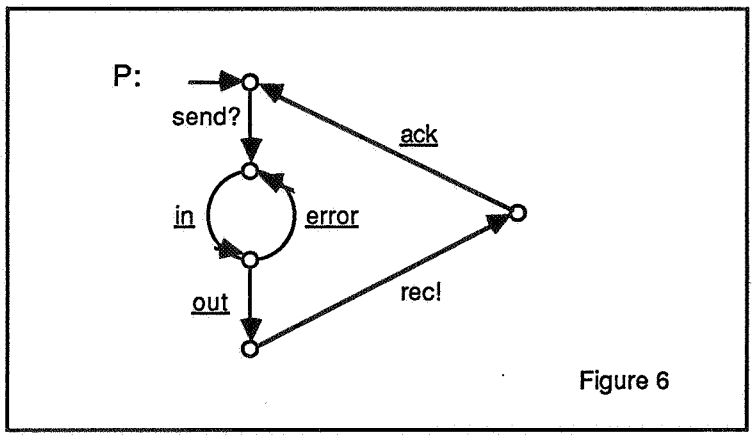



Figure 6

Here the action in denotes the result of a handshake communication between in? and in! performed through parallel composition || (see [BK 1,3]). Analogously for error, out and ack.

These communications should be treated as internal actions of the protocol, not observable from the outside. Therefore the remaining task of protocol verification is now to show that the system

$$SYS = \tau_I(P),$$

obtained from P by hiding all communications in I = {in, error, out, ack}, behaves like a reliable transmission line specified by

$$SPEC = send?.out!.SPEC,$$

i.e. to derive the equation

$$SYS = SPEC.$$

Intuitively, this verification depends on the fairness assumption that the medium M and hence the protocol P will eventually leave its cycle of errors and correctly output the current message to the receiver R. Algebraically, this assumption is treated by Koomen's Fair Abstraction Rule KFAR. Its application to $P_0$ yields

$$\tau_I(P_0) = \tau.\tau_I(out.rec!.ack.P).$$

By the axioms TI1-4 and T1 of BS, we continue

$$\begin{aligned} \tau_I(P) &= send?.\ \tau_I(P_0) \\ &= send?.\tau.\tau.rec!.\tau.\tau_I(P) \\ &= send?.rec!.\ \tau_I(P). \end{aligned}$$

Now the Recursive Specification Principle RSP yields

$$SYS = \tau_I(P) = SPEC$$

as desired.

# 4. Failure Semantics with Explicit Divergence: $FS_\Delta$.

## 4.1. The Question.

As we have seen, Koomen's Fair Abstraction Rule KFAR is an attractive means for the algebraic verification of processes that involve some notion of fairness. KFAR is justified in bisimulation semantics BS, a very discriminating semantics which preserves the full branching structure of processes. Often this seems too detailed so that we would like to "simplify" the results obtained in BS.

To illustrate this point we consider an example from [BBK 1] describing a small part of an operating system, viz. the interaction of a printer with a file handler that might crash when attempting to send its data to the printer. Let action "c" denote the occurrence of a crash and action "$d_k$" the succesful output of the k-th data item by the printer, $k \geq 0$. Then the final observable effect of the interaction printer-file is given by the process $x_0$ defined as follows:

(B) $$x_k = \tau.y_k + \tau.c$$
$$y_k = d_k.x_{k+1} + \tau.d_k.y_{k+1} + \tau.d_k.c$$

where $k \geq 0$. The process graph of $x_0$ is shown in Figure 7.

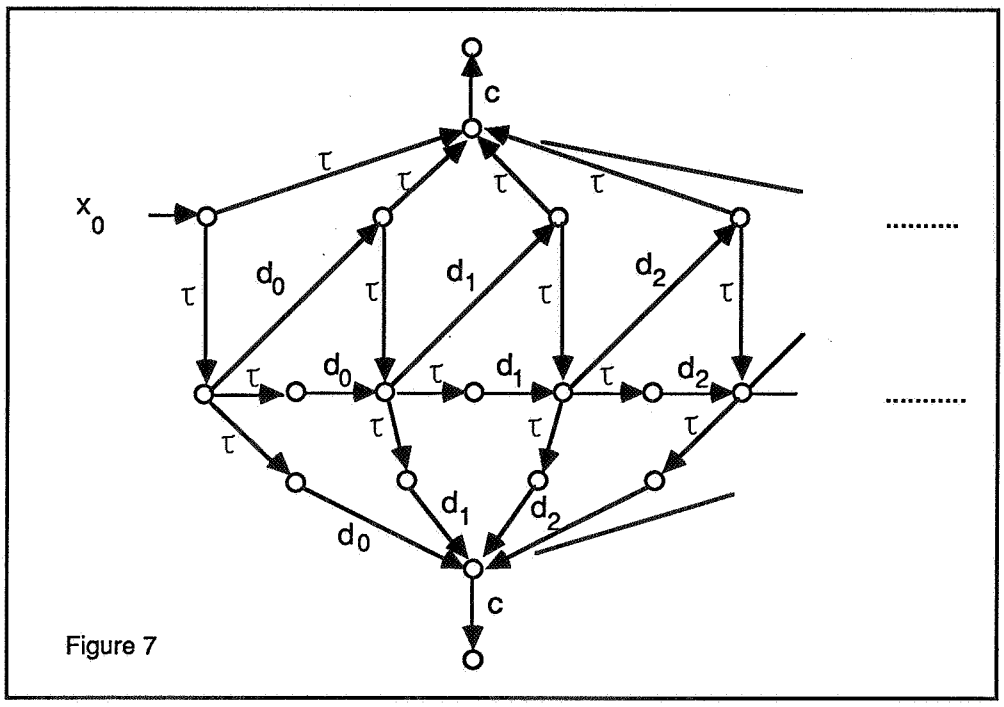



Figure 7

In the axiom system BS the equations for $x_0$ cannot be simplified any further. This fact is proved by showing that in the model $\mathcal{A}$(BS) none of the nodes in the process graph of $x_0$ can be collapsed or deleted.

However, looking at communication traces the behaviour of $x_0$ can be summarised very easily: $x_0$ outputs the data items $d_0, d_1, d_2, \ldots$ in a row, but at any moment a crash "c" can occur after which no further action happens. In general, such a linear, i.e. trace-like description of processes is often adequate (cf. [Pn]). This observation brings us to the main question of our paper:

*Can the advantages of fair abstraction be combined with the simplicity of a trace consistent, linear process semantics?*

**4.2. Towards a Solution.**

We begin with the description of a linear semantics, first with explicit divergence $\Delta$. The simplest way of introducing such a semantics would be to add the linearity axiom

$$x(y + z) = xy + xz \qquad \text{L}$$

to $BS_\Delta$. Unfortunately, we have:

PROPOSITION 3. $BS_\Delta + L$ *is trace inconsistent.*

PROOF. Clearly, the equation $a(b + \delta) = ab + a\delta$ is an instance of L. Nevertheless
$\underline{\text{trace}}(a(b + \delta)) = \underline{\text{trace}}(ab) = \{ab\surd\} \neq \{ ab\surd, a\delta\} = \underline{\text{trace}}(ab + a\delta)$. □

To achieve trace consistency, we take the more discriminating failure semantics of [BHR]. As shown in [Br, dNH, He, OH, BKO], this semantics enjoys a number of remarkable properties. For example, in [BKO] it is shown that for finite processes without $\tau$ failure semantics yields the largest trace consistent congruence. Thus failure semantics identifies as much as possible without producing any trace inconsistency. Moreover, in [dNH, He] it is shown that for (divergence free) processes failure semantics coincides with a so-called acceptance semantics obtained through a very natural idea of testing processes.

Here we axiomatise failure semantics in the presence of $\tau$ and $\Delta$. Starting from $BS_\Delta$ we add the following axioms with $\alpha,\beta$ ranging over $A \cup \{\delta,\tau\}$:

$$\alpha(\beta x + u) + \alpha(\beta y + v) = \alpha(\beta x + \beta y + u) + \alpha(\beta x + \beta y + v) \qquad \text{R1}$$

$$\tau x + y = \tau x + \tau(x + y) \qquad \text{T4}$$

$$\frac{\forall k \in \mathbb{N}: \quad x_k = i_k . x_{k+1} + y_k, \quad i_k \in I}{\tau_I(x_0) = \Delta . \tau_I(\Sigma_{k\in\mathbb{N}}\ y_k)} \qquad \text{DE}_\infty$$

The resulting axiom system we call $FS_\Delta$.

Axioms R1 and T4 are derivable from the axioms for failure and acceptance semantics in [Br]

and [dNH]. For finite processes R1 completely characterises the readiness semantics of [OH]. This is shown in [BKO] where a complete axiomatisation of that semantics is given. We therefore call R1 the *readiness axiom*. T4 augments Milner's $\tau$-laws T1-3 of bisimulation semantics; it is explicitly listed as a derived axiom in [dNH]. For finite processes T4 and R1 completely characterise the failure semantics. This is proved in [Br, BKO], but also in [dNH] for the equivalent acceptance semantics.

New is our way of dealing with divergence in connection with failure semantics. It is axiomatised in the "infinitary" Delay Rule $DE_\infty$ which extends the"periodical" version $DE_n$ in $BS_\Delta$. To obtain a finite sum in the conclusion, we apply $DE_\infty$ only in case of finitely many different $y_k$.

We examine the impact of the new axioms in $FS_\Delta$. The readiness axiom R1 is a restricted form of the linearity axiom L; indeed, for $\alpha,\beta \in A \cup \{\delta,\tau\}$ we obtain:

PROPOSITION 4. $FS_\Delta \vdash \alpha(\beta x + \beta y) = \alpha\beta x + \alpha\beta y$.

PROOF. Put $u = v = \delta$ in R1 and use axioms A3 and A6. □

Clearly, this equation is wrong under bisimulation semantics $BS_\Delta$. We demonstrate the effect of Proposition 4 by applying it to the final result of the printer-file example from above. In $FS_\Delta$ the equations (B) for $x_0$ can be simplified drastically, viz. to

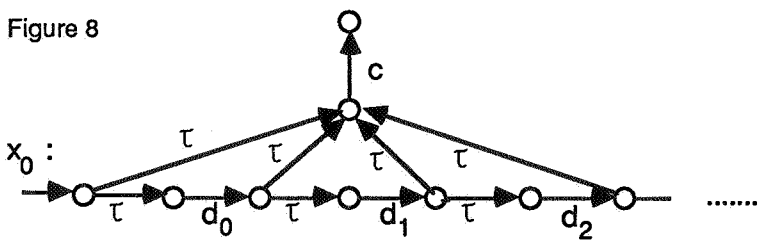
(F) $\qquad x_k = \tau.d_k.x_{k+1} + \tau.c$

where $k \geq 0$. Note that (F) results from (B) by dropping in each equation for $y_k$ the summand

$$+ \tau.d_k.y_{k+1} + \tau.d_k.c\ .$$

The simplicity of (F) is best illustrated by looking at the new process graph for $x_0$ shown in Figure 8.

Figure 8




To derive (F) formally from (B), it suffices to show for $k \geq 0$:

$FS_\Delta \vdash \tau.y_k = \tau.d_k.x_{k+1}$.

PROOF.

| | |
|---|---|
| $\tau.y_k =$ | [def.$y_k$] |
| $\tau.(d_k.x_{k+1} + \tau.d_k.y_{k+1} + \tau.d_k.c) =$ | [def. $x_{k+1}$] |
| $\tau.(d_k(\tau.y_{k+1} + \tau c) + \tau.d_k.y_{k+1} + \tau.d_k.c) =$ | [T1] |
| $\tau.(d_k(\tau.y_{k+1} + \tau c) + \tau.d_k.\tau.y_{k+1} + \tau.d_k.\tau c) =$ | [twice Prop.4] |
| $\tau.(d_k(\tau.y_{k+1} + \tau c) + \tau.d_k(\tau.y_{k+1} + \tau c)) =$ | [T2] |
| $\tau.\ \tau.d_k(\tau.y_{k+1} + \tau c) =$ | [T1] |
| $\tau.d_k(\tau.y_{k+1} + \tau c) =$ | [def. $x_{k+1}$] |
| $\tau.\ d_k.x_{k+1}$ □ | |

An immediate consequence of the Delay Rule $DE_\infty$ is the finitary rule DE(2):

$$\frac{x_0 = i_0.x_1 + y_0, \quad x_1 = i_1.x_1 + y_1, \quad i_0, i_1 \in I}{\tau_I(x_0) = \Delta.\tau_I(y_0 + y_1)} \qquad \text{DE(2)}$$

(Note that DE(2) does not follow from from the "periodical" versions $DE_n$, $n \geq 1$, of $BS_\Delta$.)

PROOF. To apply $DE_\infty$, replace the second premise of DE(2) by the infinite system of equations

$$x_k = i_k.x_{k+1} + y_1$$

where $k \geq 1$. □

With DE(2) we derive the following fact about $\Delta$:

PROPOSITION 5. (i) $FS_\Delta \vdash \Delta = \Delta\delta + \varepsilon$, (ii) $FS_\Delta \vdash \Delta(x + y) = \Delta x + \Delta y$.

PROOF. For (i) consider the system of equations

$$x_0 = ix_1 + \varepsilon$$
$$x_1 = ix_1 + \delta$$

Then

| | | |
|---|---|---|
| $\Delta$ | = | [A6,8] |
| $\Delta(\varepsilon + \delta)$ | = | [TI1-4] |
| $\Delta\tau_{\{i\}}(\varepsilon + \delta)$ | = | [DE(2)] |
| $\tau_{(i)}(x_0)$ | = | [def.$x_0$] |
| $\tau_{\{i\}}(ix_1 + \varepsilon)$ | = | [TI1-4] |
| $\tau.\tau_{\{i\}}(x_1) + \varepsilon$ | = | [$DE_1$] |
| $\tau\Delta\delta + \varepsilon$ | = | [Cor.1] |
| $\Delta\delta + \varepsilon$. | | |

Now (ii) follows immediately:

$$\Delta(x + y) = \Delta\delta + x + y = \qquad [A3]$$
$$\Delta\delta + x + \Delta\delta + y = \Delta x + \Delta y.$$

□

The failure specific linearity (ii) of Δ is remarkable in view of the rejected linearity axiom L.

### 4.3. A Model for $FS_\Delta$.

We obtain a model for the axiom system $FS_\Delta$ by taking the process graphs $\mathcal{G}$ modulo a suitable modification of the failure equivalence ≡ in [Br], called here Δ-failure equivalence $\equiv_\Delta$.

Its definition is based on the mapping $F_\Delta$ assigning to each process graph $g \in \mathcal{G}$ a set

$$\mathcal{F}_\Delta[g] \subseteq A^* \times \wp(A \cup \{\surd\}) \cup \{\tau\} \cup A^*.\{\surd\} \cup A^*.\{\Delta\}.$$

The elements of this set record the following information:
- failure pairs $(\sigma, X)$ with $\sigma \in A^*$ and $X \subseteq A \cup \{\surd\}$ record the deadlock possibilities, i.e. the moves that can be refused after σ [BHR, OH];
- the element τ indicates an initial τ-edge in g;
- elements $\sigma\surd$ record complete traces leading to an exit node and elements $\sigma\Delta$ traces leading to a divergence node or a 'subdivergence' node.

Formally, we define for a node s in a process graph g the set

$$init(s) = \{a \mid a \in A \text{ and } \exists t: s \Longrightarrow^a_g t\} \cup \{\surd \mid s \text{ is an exit node}\}.$$

Thus init(s) records the initial moves from s and the possibility of termination. Further on, a node is called *stable node* if it is not the starting point of an edge labelled by τ; otherwise it is called *unstable node*. For use only in this section, a node s is called a *subdivergence node* if it can be reached from the root by a trace σρ such that there is a divergence node t reachable by σ from the root. So in particular divergence nodes are subdivergence nodes. It is not required that the subdivergence node s is actually reachable from the divergence node t; however, after inserting a suitable 'τ-jump' (to be defined shortly) s is indeed below t in the graph.

Given a process graph $g \in \mathcal{G}$ with root r we define $\mathcal{F}_\Delta[g]$ as the least set satisfying the following conditions:

(F1) $(\sigma, X) \in \mathcal{F}_\Delta[g]$ if $r \Longrightarrow^\sigma_g s$ for some stable node s and the set $X \subseteq A \cup \{\surd\}$ is disjoint from init(s),

(F2) $\tau \in \mathcal{F}_\Delta[g]$ if r is an unstable node,

(F3) $\sigma\surd \in \mathcal{F}_\Delta[g]$ if $r \Longrightarrow^\sigma_g s$ for some exit node s,

(F4) $\sigma\Delta \in \mathcal{F}_\Delta[g]$ if $r \Longrightarrow^\sigma_g s$ for some subdivergence node s.

Now, Δ-failure equivalence $\equiv_\Delta$ on $\mathcal{G}$ is defined by

$$g \equiv_\Delta h \text{ iff } \mathcal{F}_\Delta[g] = \mathcal{F}_\Delta[h].$$

Except for its reference to stable nodes, condition F1 is as in [Br]. Conditions F2-4 are not needed in [Br] because only finite processes without termination ε and without Milner's nondeterminism operator + are studied there. In particular, condition F2 ensures, similarly to condition B4 of Δ-bisimilarity in Section 2.3, that Δ-failure equivalence is a congruence even for +. Note that a divergence node does not contribute any failure pair (σ,X) due to the stability requirement in F1. Especially remarkable is clause F4: if there 'subdivergence' is replaced by 'divergence', the resulting equivalence is not a congruence. (This observation is due to R. van Glabbeek.)

We can now state:

THEOREM 2. (i) $\equiv_\Delta$ *is a congruence on* $\mathcal{G}$. (ii) $\mathcal{G}/\equiv_\Delta \models FS_\Delta$, *i.e. modulo Δ-failure equivalence the process graphs satisfy all the axioms of* $FS_\Delta$.

Note. The proof is routine, tedious and omitted. We remark that - unlike Theorem 1 - this theorem crucially depends on the restriction that $\mathcal{G}$ contains only finitely branching graphs. □

Notation: $\mathcal{A}(FS_\Delta) = \mathcal{G}/\equiv_\Delta$. We remark that the way in which the model $\mathcal{A}(FS_\Delta)$ treats divergence differs from the full failure model [BHR, BR] (see Section 5) and from the acceptance models developed in [dNH, He]. In contrast to these models $\mathcal{A}(FS_\Delta)$ continues to record failure pairs (σ,X) even after a divergence is encountered. It is this model property which enables us to add in Section 6 fair abstraction on top of $FS_\Delta$.

In Proposition 5 we noticed the linearity of Δ in $FS_\Delta$. We can now "explain" this property using a simple process graph transformation that is valid in the model $\mathcal{A}(FS_\Delta)$. This transformation we call the τ-*jump*; it is shown in Figure 9.

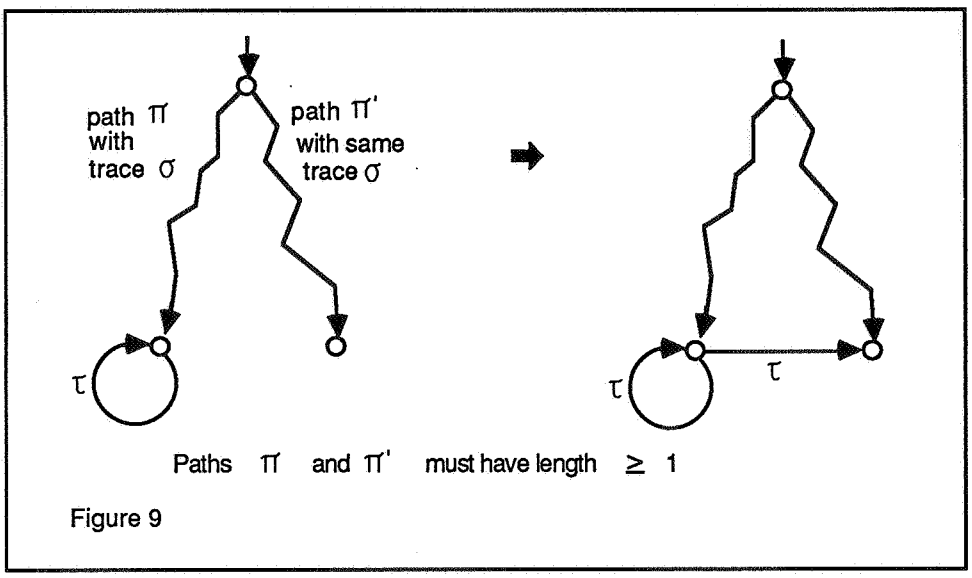



Figure 9

With the τ-jump (and Δ-bisimilarity $\triangleq_\Delta$) we derive in Figure 10 the linearity of Δ pictorially.

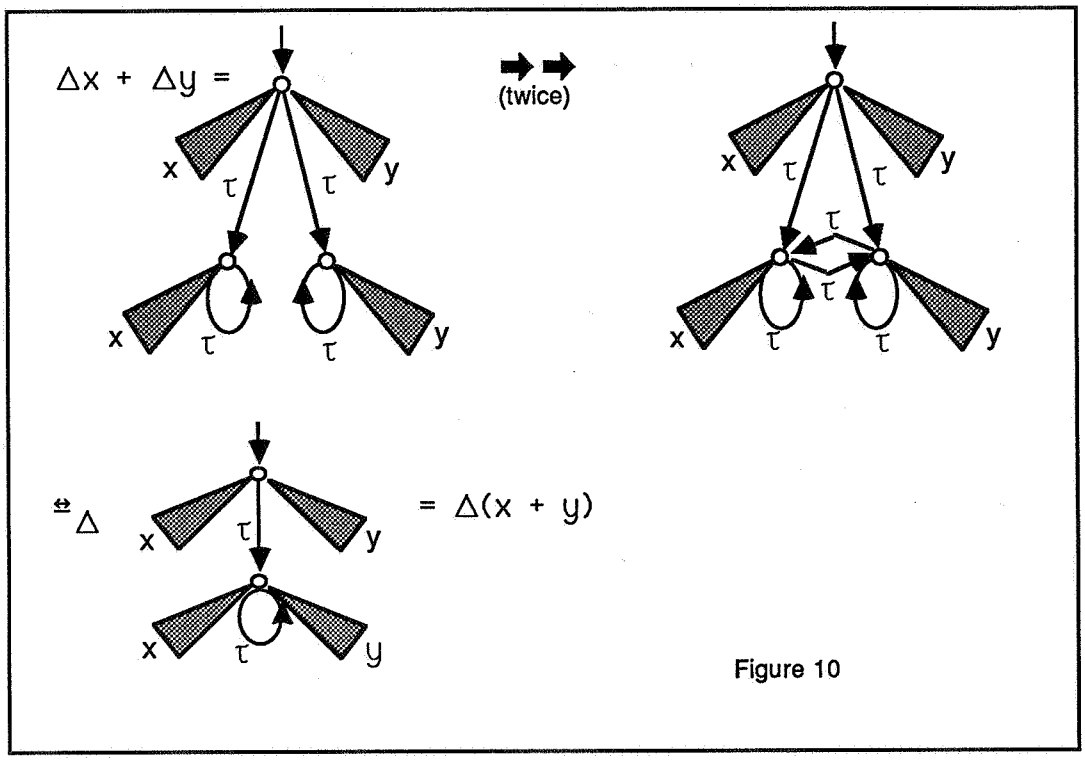



Figure 10

## 5. Failure Semantics with Catastrophic Divergence: $FS_\chi$

We now come to the main point of our question: can, as in bisimulation semantics $BS_\Delta$, the divergence $\Delta$ be abstracted away? Let us try the axiom

$$\Delta = \tau$$

that yields Koomen's Fair Abstraction Rule KFAR when added to $BS_\Delta$. Surprisingly, we can use the linearity of $\Delta$ to obtain:

PROPOSITION 6. *The system* $FS_\Delta + (\Delta = \tau)$ *is trace inconsistent.*

PROOF. By Proposition 5, $\Delta = \tau$ implies $\tau(x + y) = \tau x + \tau y$ which is trace inconsistent analogously to Proposition 3. □

Another trace inconsistency of failure semantics directly with KFAR instead of $DE_\infty$ and $\Delta = \tau$ is shown in [BKO].

So how did Brookes, Hoare and Roscoe [BHR], the inventors of failure semantics, manage? They treated divergence as being "catastrophic": all processes with an infinite $\tau$-sequence from the root are identified with the wholly arbitrary process, called CHAOS. We can express their solution by adding to $FS_\Delta$ the axiom

$$\Delta\delta = \Delta$$

which states that $\Delta$ never terminates. Call the resulting system $FS_\chi$ ($\chi$ for CHAOS).

PROPOSITION 7. (i) $FS_\chi \vdash \Delta x = \Delta$ (ii) $FS_\chi \vdash \Delta + x = \Delta$.

PROOF. $\Delta + x = \Delta\delta + x =$ [Prop. 5]
$\Delta x = \Delta\delta x =$ [A7]
$\Delta\delta = \Delta$. □

The equations (i) and (ii) characterise $\Delta$ as the process CHAOS of [BHR] that makes any distinction of the subsequent or alternative process behaviour impossible.

As a model $\mathcal{A}(FS_\chi)$ for $FS_\chi$ essentially the original failure model in [BHR, BR] or the equivalent ("must"-version of) acceptance model in [dNH] suffices.

## 6. Failure Semantics with Fair Abstraction of Unstable Divergence: FS

### 6.1. Fair Abstraction.

Thus failure semantics seems unfit for fair abstraction and to point to CHAOS. Also the different versions of acceptance semantics of [dNH, He] do not help here. Nevertheless there is a surprising solution. We can formulate a restricted fair abstraction principle:

$$\Delta\tau = \tau.$$

This axiom says that a process will never stay forever in a cycle of internal $\tau$-steps if it can be exited by another internal $\tau$-step. Since in $\Delta\tau$ the status of being divergent is itself unstable due to the $\tau$ we refer to $\Delta\tau$ as *unstable divergence* and to $\Delta\tau = \tau$ as abstraction of unstable divergence.

Let FS be the axiom system $FS_\Delta + (\Delta\tau = \tau)$. In FS the following Fair Abstraction Rule of Unstable Divergence $KFAR^-$ is derivable:

$$\frac{\begin{array}{l}\forall k \in \mathbb{N}: \quad x_k = i_k.x_{k+1} + y_k, \quad i_k \in I \\ \exists\, k \in \mathbb{N}\ \exists\, i \in I\ \exists z:\ y_k = i.z\end{array}}{\tau_I(x_0) = \tau.\tau_I(\Sigma_{k\in\mathbb{N}}\ y_k)} \qquad KFAR^-$$

PROOF. Applying the rule $DE_\infty$ of $FS_\Delta$ to the first premise of $KFAR^-$ yields

$$\tau_I(x_0) = \Delta.\tau_I(\Sigma_{k\in\mathbb{N}}\ y_k).$$

Using the second premise of KFAR$^-$ and the failure specific axioms T4 and R1 (more precisely its consequence stated in Proposition 4) we derive further

$$\tau_I(x_0) = \Delta\tau.\tau_I(\Sigma_{k\in\mathbb{N}}\ y_k).$$

A final application of $\Delta\tau = \tau$ yields the consequence of KFAR$^-$. □

**6.2. Protocol Verification in FS.**

An inspection of the algebraic verification of the Alternating Bit [BK 3] and Sliding Window Protocols [Va], which was done in the setting of bisimulation semantics using KFAR, shows that this verification is also possible in the system FS with KFAR$^-$. Thus KFAR$^-$ is an interesting alternative to KFAR.

We demonstrate this here for the idealised protocol P of Section 3.2. Recall that P satisfies the equations

$$\begin{aligned} P &= \mathsf{send?}.P_0 \\ P_0 &= \underline{\mathsf{in}}.P_1 \\ P_1 &= \underline{\mathsf{error}}.P_0 + \underline{\mathsf{out}}.\mathsf{rec!}.\underline{\mathsf{ack}}.P \end{aligned}$$

and that in order to complete the verification it suffices to show

$$(*) \qquad \tau_I(P) = \mathsf{send?}.\mathsf{rec!}.\ \tau_I(P)$$

where $I = \{\underline{\mathsf{in}}, \underline{\mathsf{error}}, \underline{\mathsf{out}}, \underline{\mathsf{ack}}\}$.

Consider the equations for $P_0$ and $P_1$. Since the second summand of $P_1$ starts with the action $\underline{\mathsf{out}} \in I$, the rule KFAR$^-$ is applicable and yields:

$$\tau_I(P_0) = \tau.\tau_I(\underline{\mathsf{out}}.\mathsf{rec!}.\underline{\mathsf{ack}}.P).$$

Now we can continue as in Section 3.2:

$$\begin{aligned} \tau_I(P) &= \mathsf{send?}.\ \tau_I(P_0) \\ &= \mathsf{send?}.\ \tau.\tau_I(\underline{\mathsf{out}}.\mathsf{rec!}.\underline{\mathsf{ack}}.P) \\ &= \mathsf{send?}.\mathsf{rec!}.\ \tau_I(P). \end{aligned}$$

Hence (*) holds in the axiom system FS.

**6.3. A Model for FS.**

We first introduce a failure semantics $\mathcal{F}^*$ that assigns to every process graph $g \in \mathcal{G}$ a set

$$\mathcal{F}^*[g] \subseteq A^* \times \wp(A \cup \{\surd\}) \ \cup\ \{\tau\} \ \cup\ A^*.\{\surd\} \ \cup\ A^*.$$

Compared with $\mathcal{F}_\Delta[g]$ no divergence elements $\sigma\Delta$ appear; instead all (partial) traces $\sigma \in A^*$ are recorded. $\mathcal{F}^*[g]$ is defined as the least set satisfying the conditions F1-3 of $\mathcal{F}_\Delta[g]$ but with F4 replaced by the new condition

(F4*) $\quad \sigma \in \mathcal{F}^*[g] \quad$ if $r \Longrightarrow^{\sigma}_{g} s$ for the root r and some node s of g.

Defining the new failure equivalence $\equiv^*$ on the graph domain $\mathcal{G}$ by

$$g \equiv^* h \text{ iff } \mathcal{F}^*[g] = \mathcal{F}^*[h],$$

we take as model $\mathcal{A}(FS) = \mathcal{G}/\equiv^*$.

THEOREM 3. (i) $\equiv^*$ *is a congruence on* $\mathcal{G}$, (ii) $\mathcal{A}(FS) \models FS$.

We show only, in Figure 11, how the crucial laws $\Delta\tau = \tau$ and $\Delta \neq \tau$ are realised in $\mathcal{A}(FS)$.

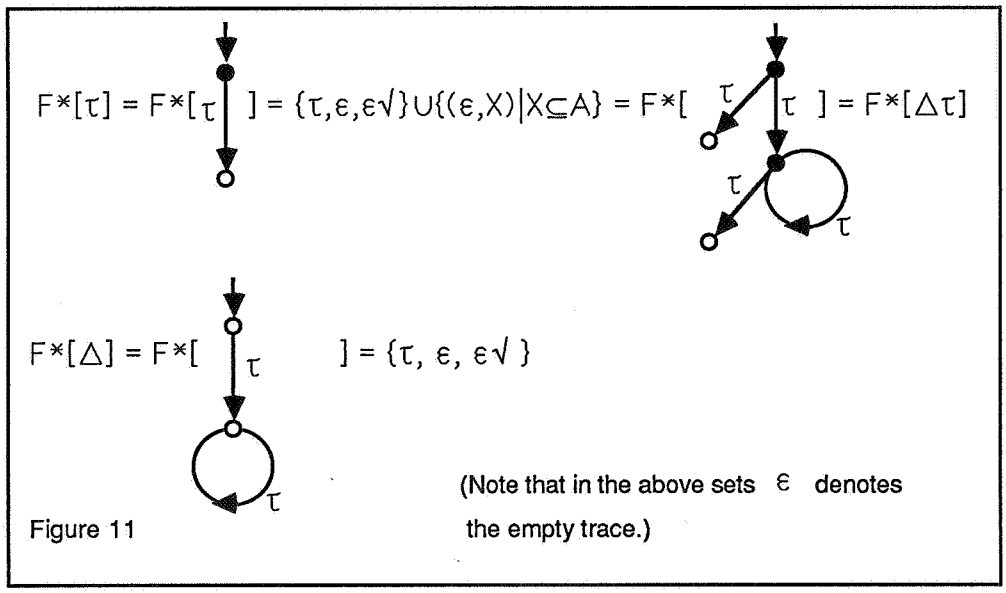



Figure 11

(Note that in the above sets $\varepsilon$ denotes the empty trace.)

Clearly, these laws are not realised in the failure semantics of [BHR, BR] or any of the acceptance semantics of [dNH, He].

## 7. Conclusion

The axiom systems BS and FS appear to be attractive for algebraic process verification. The advantage of the "branching system" BS is that the full abstraction rule KFAR is available. On the other hand, verifications in BS can be very tedious because so many distinctions are made -

unnecessarily many the supporters of the "linear approach" FS might say. The simplicity of FS over BS is best illustrated by looking at the set Char of finite process terms which are built exclusively from the characteristic processes $\varepsilon$, $\delta$, $\tau$, $\Delta$ by means of . and +.

PROPOSITION 8 [KV]. *Under the bisimulation semantics* BS *with fair abstraction* Char *contains infinitely many semantically different processes.*

PROPOSITION 9. *Under the failure semantics* FS *with fair abstraction of unstable divergence every process in* Char *is semantically equal to one of the seven processes:* $\delta$, $\varepsilon$, $\Delta\delta$, $\Delta$, $\tau$, $\tau\delta$, $\tau+\tau\delta$. *These seven processes cannot be collapsed any further without introducing a trace inconsistency.*

A disadvantage of the above axiom systems $BS_\Delta$, BS, $FS_\Delta$, $FS_\chi$ and FS is that each of them treats all occurrences of divergence $\Delta$ in a process in the same way. Consequently the fairness assumptions represented by KFAR or $KFAR^-$ are only of global nature; in a process built up from several components it is not clear which one is responsible for this global fairness. In our future work we intend to investigate axiom systems where different forms of divergence coexist. An interesting question is whether this idea allows us to deal purely algebraically, i.e. by means of equations only, with local fairness assumptions. If so, this approach would be complementary to the work of Parrow where local fairness is expressed using temporal logic formulas in addition to the algebraic framework [Par].

**Acknowledgement.** We would like to thank R. van Glabbeek for pointing out to us the difference between our bisimilarity with explicit divergence and the related notion of Milner in [Mi 2].

## References

[BBK 1] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Rep. CS-R8503, CWI, Amsterdam 1985.

[BBK 2] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, *On the consistency of Koomen's Fair Abstraction Rule*, Rep. CS-R8511, CWI, Amsterdam 1985.

[BMOZ] J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Transition systems, infinitary languages and the semantics of uniform concurrency*, in: Proc. 17th ACM STOC, Providence, R.I., 1985.

[BK 1] J.A. Bergstra, J.W. Klop, *Algebra of communicating processes*, to appear in: J.W. de Bakker, M. Hazewinkel, J.K. Lenstra, Eds., Proc. CWI Symp. Math. and Comp. Sci., North-Holland, Amsterdam.

[BK 2] J.A. Bergstra, J.W. Klop, *Process algebra for synchronous communication*, Inform. and Control 60 (1984) 109-137.

[BK 3] J.A. Bergstra, J.W. Klop, *Verification of an alternating bit protocol by means of process algebra*, Rep. CS-R8404, CWI, Amsterdam 1984.

[BK 4] J.A. Bergstra, J.W. Klop, *Algebra of communicating processes with abstraction*, TCS 37 (1985) 77-121.

[BKO] J.A. Bergstra, J.W. Klop, E.-R. Olderog, *Readies and failures in the algebra of communicating processes*, Rep. CS-R8523, CWI, Amsterdam 1985.

[vB] G. v. Bochmann, *Concepts of distributed systems design*, (Springer-Verlag, Berlin, 1983).

[Br] S.D. Brookes, *On the relationship of CCS and CSP*, in: J. Díaz, Ed., Proc. 10th ICALP, Springer-LNCS 154 (1983) 83-96.

[BHR] S.D. Brookes, C.A.R. Hoare, A.W. Roscoe, *A theory of communicating sequential processes*, J.ACM 31 (1984) 560-599.

[BR] S.D. Brookes, A.W. Roscoe, *An improved model for communicating sequential processes*, in: S.D. Brookes, A.W. Roscoe, G. Winskel, Eds., Proc. NSF-SERC Seminar on Concurrency, Springer-LNCS 197 (1985).

[Ha] B. Hailpern, *Verifying concurrent processes using temporal logic*, Springer-LNCS 129, 1982.

[He] M. Hennessy, *Acceptance trees*, J.ACM 32 (1985).

[Ho] C.A.R. Hoare, *Communicating sequential processes*, (Prentice-Hall, London, 1985).

[Ko] C.J. Koomen, *Algebraic specification and verification of communication protocols*, SCP 5 (1985) 1-36.

[KV] C.P.J. Koymans, J.L.M. Vrancken, personal communication.

[Mi 1] R. Milner, *A calculus of communicating systems*, Springer-LNCS 92, 1980.

[Mi 2] R. Milner, *A modal characterisation of observable machine-behaviour*, in: E. Astesiano, C. Böhm (Eds.), Proc. 6th CAAP, Springer-LNCS 112 (1981) 25-34.

[Mi 3] R. Milner, *Calculi for synchrony and asynchrony*, TCS 25 (1983) 267-310.

[dNH] R. de Nicola, M. Hennessy, *Testing equivalences for processes*, TCS 34 (1984) 83-134.

[OH] E.-R. Olderog, C.A.R. Hoare, *Specification-oriented semantics of communicating processes*, Acta Informatica 23 (1986) 9-66.

[Pa] D. Park, *Concurrency and automata on infinite sequences*, in: P. Deussen, Ed., Proc. 5th GI Conf. on Theoret. Comp. Sci., Springer-LNCS 104 (1981).

[Par] J. Parrow, *Fairness properties in process algebra - with applications in communication protocol verification*, Ph.D. thesis, Dept. of Comp. Sci., Uppsala Univ., 1985.

[Pn] A. Pnueli, *Linear and branching structures in the semantics and logics of reactive systems*, in: W. Brauer, Ed., Proc. 12th ICALP, Springer-LNCS 194 (1985) 15-32.

[Ta] A.S. Tanenbaum, *Computer networks*, (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[Va] F. Vaandrager, *Verification of two communication protocols by means of process algebra*, Rep. CS-R8608, CWI, Amsterdam 1986.