Ready-trace semantics for concrete process algebra with the priority operator

Ready-Trace Semantics for Concrete Process Algebra with the Priority Operator

J. C. M. BAETEN*†, J. A. BERGSTRA†‡ AND J. W. KLOP§

† *Department of Computer Science, University of Amsterdam*

‡ *Department of Philosophy, University of Utrecht*

§ *Centre for Mathematics and Computer Science, Amsterdam*

We consider a process semantics intermediate between bi-simulation semantics and readiness semantics, called here ready-trace semantics. The advantage of this semantics is that, while retaining the simplicity of readiness semantics, it is still possible to augment this process model with the mechanism of atomic actions with priority (the θ operator). It is shown that in readiness semantics and* a fortiori *in failure semantics such an extension with θ is impossible. Ready-trace semantics is considered here in the simple setting of concrete process algebra, that is: without abstraction (no silent moves), moreover for finite processes only. For such finite processes without silent moves a complete axiomatisation of ready-trace semantics is given via the method of process graph transformations.

Received March 1986, revised August 1986

1. INTRODUCTION

In Baeten, Bergstra and Klop[1] an operator θ is introduced, in the setting of bi-simulation semantics, which introduces priorities on the atomic steps in a process. This priority operator is useful in process specification; for example one can model interrupt mechanisms with it (see Ref. 1). In Bergstra, Klop and Olderog[2] a complete axiomatisation for finite processes with communication, but without silent moves, is given for readiness semantics and failure semantics.

Now the starting point for this paper is the question whether θ can consistently be added to readiness and failure semantics as expounded in Ref. 2. This is not obvious, since readiness and failure semantics equate many more processes than bi-simulation semantics does. Indeed it turns out that θ and readiness or failure semantics are inconsistent (Section 1).

The question next considered is whether there is a process semantics 'close to' readiness and failure semantics to which θ can be consistently added. It turns out that there is a very natural semantics with this property: *RTS*, ready-trace semantics, which is interesting for its own sake. In Pnueli[8] *RTS* is called 'barbed semantics'. We give a complete axiomatisation for finite τ-less processes under this semantics, as well as a complete axiomatisation (RTS_θ) which, moreover, takes θ into account. The method of proof (to obtain the completeness results) is via process graph transformations which enjoy the termination and confluency property, as in Ref. 2.

This paper can be read independently, but it is useful to have seen Refs 1 and 2. Some general references are: for bi-simulation semantics, Milner;[6] for readiness semantics, Olderog and Hoare;[7] for failure semantics, Brookes, Hoare and Roscoe,[5] and for a connection between bi-simulation and failure semantics, Brookes.[4] A more complete list of references can be found in Refs 1 and 2.

* To whom correspondence should be addressed.

2. THE INCOMPATIBILITY OF READINESS AND FAILURE SEMANTICS WITH THE PRIORITY OPERATOR

We start with the simple demonstration of the fact that it is impossible to extend readiness and (*a fortiori*) failure semantics with the priority operator θ, as this observation is one of the primary motivations for the introduction of the ready-trace semantics below. First we will make precise the different concepts involved in this observation. We recall the following facts from Bergstra, Klop and Olderog.[2]

2.1. *Theorem*

The axiom systems BPA_δ, $BPA_\delta+R1, 2$, $BPA_\delta+R1, 2+S$ in Table 1 are complete axiomatisations of bi-simulation semantics, readiness semantics and failure semantics, respectively, on finite processes (over alphabet A without τ).

Comments

Process algebra starts from a collection of given objects A, called atomic actions, atoms or steps. These actions

Table 1

BPA_δ	$x+y=y+x$	$A1$
	$(x+y)+z=x+(y+z)$	$A2$
	$x+x=x$	$A3$
	$(x+y)z=xz+yz$	$A4$
	$(xy)z=x(yz)$	$A5$
	$\delta+x=x$	$A6$
	$\delta x=\delta$	$A7$
$BPA_\delta+1, 2$	$a(bx+u)+a(by+v)=a(bx+by+u)+a(bx+by+v)$	$R1$
	$a(b+u)+a(by+v)=a(b+by+u)+a(b+by+v)$	$R2$
$BPA_\delta+R1, 2+S$	$ax+a(y+z)=ax+a(x+y)+a(y+z)$	S

are taken to be indivisible, usually have no duration and form the basic building blocks of our systems. The two compositional operators we consider here are . , denoting sequential composition, and + for alternative composition. If x and y are two processes, then $x.y$ is the process that starts the execution of y after the completion of x, and $x+y$ is the process that chooses either x or y and executes the chosen process (not the other one). Each time a choice is made, we choose from a set of alternatives. We do not specify whether the choice is made by the process itself, or by the environment. We leave out . and brackets as in regular algebra, so $xy+z$ means $(x.y)+z$. On intuitive grounds $x(y+z)$ and $xy+xz$ present different mechanisms (the moment of choice is different), and therefore an axiom $x(y+z) = xy+xz$ is not included.

We have a special constant δ denoting deadlock, the acknowledgement of a process that it cannot do anything any more, the absence of any alternative. $A_\delta = A \cup \{\delta\}$. The variables a, b range over A_δ and x, y, z, u, v range over all processes.

The notions of bi-simulation semantics, readiness semantics and failure semantics are (essentially) standard and well known; for details of their definitions referring to the present situation where two termination possibilities exist (namely a trace may end in δ, unsuccessfully, or in a proper action $\in A$, successfully) we refer to Ref. 2.

2.1.1. *Remark*

In Ref. 2 a generalisation of Theorem 2.1 is proved, where parallel operators and communication are also present. For the purpose of this section we do not need these. In Section 3 and later we will consider communication as well (i.e. BPA_δ is replaced by ACP).

2.2. In Ref. 1 the *priority operator* θ was introduced, enabling one to indicate the priority of some actions in a choice (a sum) over others. 'Priorities' are given by a partial order $>$ on A_δ where δ is always the least element. So if the ordering $a > b$ is adopted, then

$$\theta(ax+by+z) = \theta(ax+z).$$

Another property of θ is that

$$\theta(ax) = a\theta(x), \quad \text{and} \quad \theta(ax+ay) = a\theta(x)+a\theta(y), \text{ etc.}$$

Below we will give a complete (and even finite) axiomatisation of θ, but for the time being these properties of θ suffice. As shown in Ref. 1, θ is vital for modelling in process algebra features such as interrupt mechanisms. In Ref. 1 the operator θ is introduced in the setting of bi-simulation semantics. It is hence an obvious question whether such a natural operator can 'consistently' be added to readiness and failure semantics. Here we should explain what is mean by 'consistently': as an ultimate criterion for consistency of a process axiomatisation T we require that T does not derive an equation $t_1 = t_2$ between finite closed process expressions such that $trace(t_1) \neq trace(t_2)$. Here $trace(t)$ is defined in such a way that termination δs are visible, e.g. $trace(a+b\delta) = \{a, b\delta\}$. However, $trace(a+\delta) = \{a\}$, since $a+\delta = a$ (cf. axiom $A6$ in Table 1). For a precise definition (which moreover involves τ-steps) see Ref. 3.

The definite answer to the question just raised is *negative*, as the following counterexamples show.

2.3. *Proposition*

Failure semantics with the priority operator is inconsistent. In particular, $BPA_\delta+R1, 2+S+\theta$ is inconsistent.

Proof

Consider process expressions

$$t_1: ab+a(c+d)$$
$$t_2: ab+a(c+d)+a(b+c).$$

According to Table 1 (axiom S), $BPA_\delta+R1, 2+S+\theta \vdash t_1 = t_2$. Hence in this system $BPA_\delta+R1, 2+S+\theta$ we have $\theta(t_1) = \theta(t_2)$. We adopt the following priority of atoms: $b < c < d$. Now w.r.t. this ordering:

$$\theta(t_1) = a\theta(b)+a\theta(c+d) = ab+ad$$
$$\theta(t_2) = a\theta(b)+a\theta(c+d)+a\theta(b+c) = ab+ad+ac.$$

So in the axiom system under consideration we derive the equality of two expressions with different trace set. □

In fact, the previous proposition is strengthened by the following, which shows that the inconsistency is already obtainable in readiness semantics.

2.4. *Proposition*

Readiness semantics with the priority operator is inconsistent. In particular, $BPA_\delta+R1, 2+\theta$ is inconsistent.

Proof

Let

$$t_1 \equiv a(bc+d)+a(be+f)$$
$$t_2 \equiv a(be+d)+a(bc+f).$$

Adopt the priorities $d > b > f$. Now $BPA_\delta+R1, 2+\theta \vdash t_1 = t_2$ (using $R1$), but

$$\theta(t_1) = ad+abe$$
$$\theta(t_2) = ad+abc.$$ □

2.5. *Remark*

The failure of readiness semantics and failure semantics to describe the priority operator can be motivated as follows: when a process is observed to have performed some action a, then one can deduce that at that point it was unable to perform any action b of greater priority. Thus the presence of the priority operator makes visible information about possible-but-not-chosen actions at each point throughout any given trace of the process. Readiness and failure semantics only provide such information at the end of any given trace. In the next section we shall define a semantics, which will preserve this extra information.

3. READY-TRACE SEMANTICS: A MODEL OF FINITE PROCESSES

We shall now introduce *ready-trace equivalence* on processes, which will be in this paper finite and τ-less.

3.1. *Definition*

Let $\mathbb{H}$ be the domain of finite acyclic process graphs with edges labelled by elements from A_δ. The graphs $g \in \mathbb{H}$ will be supposed to be in *δ-normal form*, i.e. δ-steps may only

occur at the end of branches of g and may have no alternatives.

On $\mathbb{H}$ we define operations $+, ., \|, \lfloor\!\lfloor, |, \partial_H$ as defined in Bergstra, Klop and Olderog[2]. These definitions are assumed to be known in this paper.

Some remarks about the new operators.

$\|$ is the parallel composition operator, called merge. The merge of processes x and y will interleave the actions of x and y, except for the communication actions. In $x\|y$, we can either do a step from x, or a step from y, or x and y both synchronously perform an action, which together make up a new action, the communication action. In order to give a finite axiomatisation of merge, we use two auxiliary operators $\lfloor\!\lfloor$ (left-merge) and $|$ (communication merge). Thus, $x \lfloor\!\lfloor y$ is $x\|y$, but with the restriction that the first step comes from x, and $x|y$ is $x\|y$ with a communication step as the first step. Finally, we have the encapsulation operator ∂_H. Here H is a set of atoms, and ∂_H blocks those actions, renames them into δ. The operator ∂_H can be used to encapsulate a process, i.e. to block communications with the environment.

For the same of completeness we include here the definition of the *ready set* $\mathscr{R}[\![g]\!]$ of $g \in \mathbb{H}$:

3.2. *Definition*

Let σ vary over A^*, the set of words over the action alphabet A (not A_δ). The empty word is λ. Let $g \in \mathbb{H}$. Then the *ready set* of g, notation: $\mathscr{R}[\![g]\!]$, is the least set satisfying the following clause:

$$\text{for all } \sigma \in A^*, \quad g \xrightarrow{\sigma} h \text{ implies } (\sigma, I(h)) \in \mathscr{R}[\![g]\!].$$

Here $g \xrightarrow{\sigma} h$ (h is a *derivation* of g via σ) if there is a path, determining the word σ, from the root of g to the root of the subgraph h.

Further, $I(h)$ is the set of *initial steps* of h; if h is a single δ-step then $I(h) = \varnothing$ and if h is $\mathbb{O}$, the zero graph without edges, $I(h) = I(\mathbb{O}) = \{\varepsilon\}$. ($\varepsilon$ is a formal symbol denoting successful termination.)

3.3. *Example*

If g is the process graph in Fig. 1:

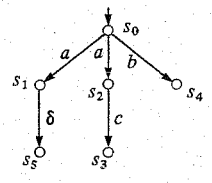

Figure 1.

then

$$\mathscr{R}[\![g]\!] = \{(\lambda, \{a, b\}), (a, \varnothing), (a, \{c\}), (ac, \{\varepsilon\}), (b, \{\varepsilon\})\}.$$

(The displayed contributions in $\mathscr{R}[\![g]\!]$ are yielded by nodes, respectively, s_0, s_1, s_2, s_3, s_4. Note that s_5 gives no contribution.)

3.4. *Definition*

$g, h \in \mathbb{H}$ are *ready equivalent* if $\mathscr{R}[\![g]\!] = \mathscr{R}[\![h]\!]$; notation $g \equiv_{\mathscr{R}} h$.

It is proved in Bergstra, Klop and Olderog[2] that ready equivalence is a congruence w.r.t. the operations $+, ., \|, \lfloor\!\lfloor, |, \partial_H$; when restricting ourselves to $+, .$ we have the isomorphism

$$\mathbb{H}(+, .)/\equiv_{\mathscr{R}} \cong I(BPA_\delta + P1, 2)$$

where $I(-)$ is the initial algebra of $-$. (This is part of the contents of Theorem 2.1 above.)

We now turn to ready-trace semantics (*RTS*); here fewer processes (process expressions) are identified than in ready semantics (*RS*), as explained above. The essential difference is that while *RS* is based on the notion of ready-pair (σ, X) (see Fig. 2(*a*)), *RTS* is based on the notion of a *ready trace* (see Fig. 2(*b*)), where also the 'intermediate' ready sets X_i along trace σ are given.

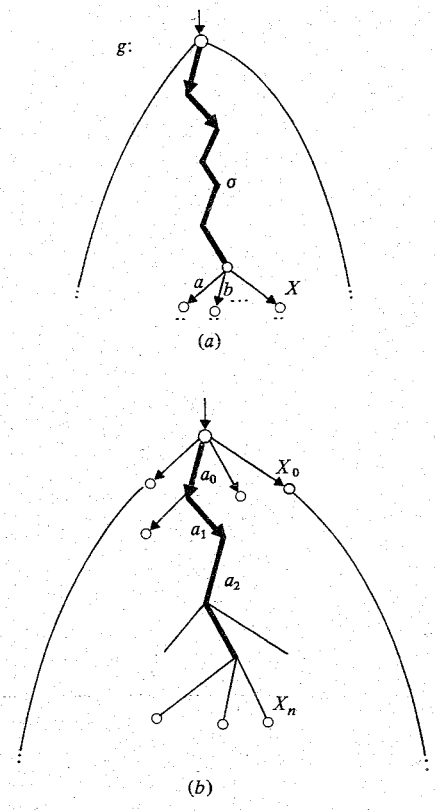

Figure 2.

3.5. *Definition*

(i) Let $g \in \mathbb{H}$. A *path* π in g, starting from the root s_0 of g, is an alternating sequence of nodes of g and labelled edges in g:

$$\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$$

with $a_i \in A$ (not A_δ). Here $n \geqslant 0$; if $n = 0$ we have the empty path. All paths in this paper will start from the root, so we omit that qualification. A path is *maximal* if it cannot be prolonged, that is, either ending in a terminal node ($\mathbb{O}$) of in the initial node of a δ-step.

(ii) If $s \in NODES(g)$, $(g)_s$ will be the *subgraph* of g with root s and consisting of all labelled edges accessible from s.

(iii) As before, $I(g)$ is the set of initial steps of g, with $I(\mathbb{O}) = \{\varepsilon\}$ and $I(\delta) = \varnothing$. Instead of $I((g)_s)$ we write just $I(s)$.

3.6. *Definition*

(i) Let $g \in \mathbb{H}$ and $\pi = s_0 \xrightarrow{a_0} \dots \xrightarrow{a_{n-1}} s_n$ be a path in g. Then $rt(\pi)$, the *ready trace corresponding to* π, is the alternating sequence of ready sets $I(s_i)$ and steps a_i:

$$I(s_0), a_0, I(s_1), a_1, \dots, a_{n-1}, I(s_n)$$

($n \geqslant 0$). We shall sometimes use the notation $(\sigma; \vec{X})$ for such a ready trace where $\sigma = a_0 a_1 \dots a_{n-1}$ and $\vec{X} = I(s_0), I(s_1), \dots, I(s_n)$. The ready trace corresponding to the empty path of g is just $I(g)$ or in the $(\sigma; \vec{X})$ notation, $(\lambda, I(g))$.

(ii) The *ready trace set* of g, notation $\mathscr{RT}[\![g]\!]$, is

$$\{rt(\pi) \mid \pi \text{ a path in } g, \text{ starting from the root}\}.$$

(iii) $g \equiv_{\mathscr{RT}} h$ if $\mathscr{RT}[\![g]\!] = \mathscr{RT}[\![h]\!]$; in words: g, h are *ready-trace equivalent*.

3.7. *Example*

(i) Let g, h be as in Fig. 3.

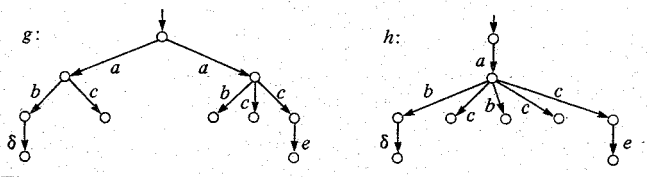

Figure 3.

Then $g \equiv_{\mathscr{RT}} h$.

(ii) g, h as in Fig. 4 are *not* ready-trace equivalent (cf. the counter-example in Proposition 2.4).

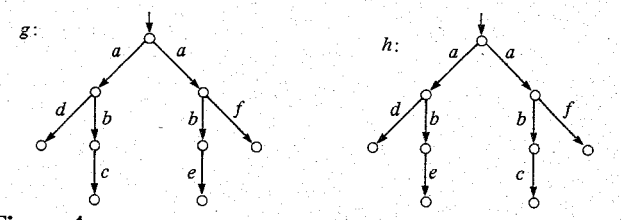

Figure 4.

Namely, $\mathscr{RT}[\![g]\!]$ contains the ready trace $\{a\}, a, \{d, b\}, b, \{c\}, c, \{\varepsilon\}$ which is not present in $\mathscr{RT}[\![h]\!]$.

3.8. *Remark*

In fact, in the present setting of finite acyclic process graphs, it would have been sufficient to consider for the definition of $\equiv_{\mathscr{RT}}$ only ready traces corresponding to *maximal* paths π. The present definition, which includes also 'prefixes' of such ready traces, anticipates working with *infinite* processes – which we shall not do in this paper.

3.9. *Remark*

A convenient 'intuition' about a ready trace is this: Imagine an interactive session with a machine as follows. At the start of the session the machine presents the user a menu of all possible actions which the user may perform ($I(g)$); one of these is chosen (a_0), whereupon the machine again flashes the menu of the options in that state ($I(s_1)$), and so on. Any moment the user may end the session, that is, leave the machine which has on its screen the last menu ($I(s_n)$). So a ready trace is a record of such a session.

We will prove in the sequel that $\equiv_{\mathscr{RT}}$ is a *congruence* on $\mathbb{H}$ w.r.t. $+$, $.$, $\|$, $\lfloor\!\lfloor$, $\lfloor$, ∂_H, and also w.r.t. the priority operator θ which will be defined now.

3.10. *Definition*

Let a partial order $<$ on A_δ be given such that $\delta < a$ for all $a \in A$. Then $\theta_<$, or θ for short, is defined on $\mathbb{H}$ as follows:

$\theta(g)$ is the process graph arising from g by

(i) erasing all edges leaving node s which have a label a majorised by label b of some other edge leaving s;

(ii) discarding all parts of g which have thus become disconnected.

3.11. *Example*

Let $a > b$ and $a > c$. Then for p as in Fig. 5(a), $\theta(p)$ is as in Fig. 5(b):

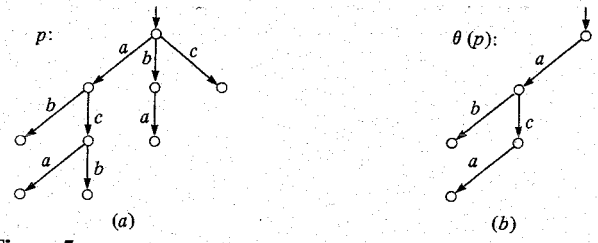

Figure 5.

4. PROCESS GRAPH TRANSFORMATIONS FOR READY-TRACE SEMANTICS

In order to prove that $\equiv_{\mathscr{RT}}$ is a congruence on $\mathbb{H}$, and to derive the completeness result in Theorem 5.2 below, we will introduce three process graph transformations of which the first two (already used in Bergstra, Klop & Olderog[2] are specific for bi-simulation semantics and the third is specific for *RTS*.

4.1. *The transformations double edge, sharing, narrowing*

(i) *Double edge*. This process graph transformation step removes in a 'double edge' $\circ \overset{a}{\underset{a}{\rightleftarrows}} \circ$ ($a \in A_\delta$) one of the edges.

Notation: $g \underset{[i]}{\Longrightarrow} h$

(ii) *Sharing*. Suppose $g \in \mathbb{H}$ contains two nodes s, t determining identical subgraphs $(g)_s$, $(g)_t$. Then the nodes s, t may be identified.

Notation: $g \underset{[ii]}{\Longrightarrow} h$.

(iii) *Narrowing*. If $g \in \mathbb{H}$ contains a part as in Fig. 6(a) this may be replaced by the part as in Fig. 6(b). More precisely, a new node r is created together with edges as in Fig. 6(b), and the old a-edges to s, t are discarded. The nodes s, t and the b_i-edges leaving them (in Fig. 6(a)) are

not discarded since s, t may have other incoming edges. (If not, then s, t are inaccessible from the root and disappear.)

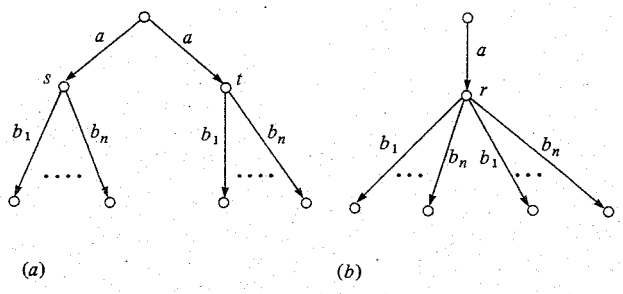

Figure 6.

Here $I(s) = I(t) = \{b_1, \ldots, b_n\}$; the b_i may have multiple occurrences among the labels of edges leaving nodes s, t.

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

4.2. *Notation*

$\Longrightarrow$ is $\underset{[i]}{\Longrightarrow} \cup \underset{[ii]}{\Longrightarrow} \cup \underset{[iii]}{\Longrightarrow}$; $\Longrightarrow\!\!\!\!\!\Rightarrow$ is the transitive reflexive closure of $\Longrightarrow$, and $\Longleftrightarrow\!\!\!\!\!\Rightarrow$ is the equivalence relation generated by $\Longrightarrow$.

4.3. *Example*

(i)

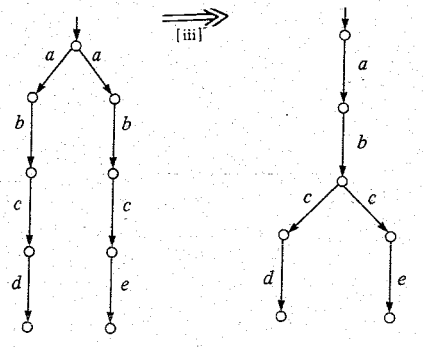

Figure 7.

(ii)

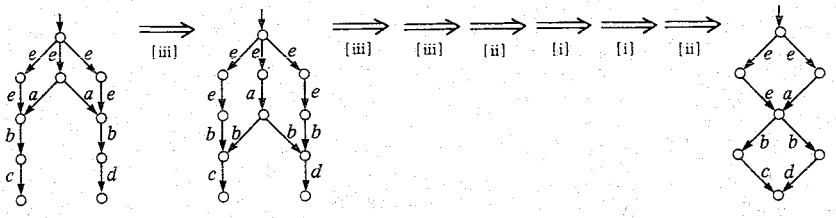

Figure 8.

4.4. *Proposition*

Let $g, h \in \mathbb{H}$. Then graph reduction is sound w.r.t. ready-trace equivalence, i.e.

$$g \Longrightarrow\!\!\!\!\!\Rightarrow h \text{ implies } g \equiv_{\mathcal{RT}} h.$$

Proof

It is simple to check that each of the three graph reductions keeps the ready trace set invariant. □

We will now establish the completeness of the graph transformations.

4.5. *Definition*

$g \in \mathbb{H}$ is in *ready trace normal form* (rt-normal form) if none of the graph reductions is applicable to g.

4.6. *Proposition*

Every graph reduction $g \Longrightarrow g' \Longrightarrow g'' \Longrightarrow \ldots$ must end eventually in a rt-normal form.

Proof

Let $T(g)$ be the tree obtained from g by unsharing. Let card(g) be the number of nodes in $T(g)$. Then transformations of type [iii] have the effect of decreasing card (g), while types [i], [ii] do not increase card (g). Since [i], [ii]-transformations clearly must end eventually, this proves the proposition.

(Note that the detour via $T(g)$ is necessary since a type [iii] transformation may increase the number of nodes in g, as Example 4.3(ii) shows: the second graph has one node more than the first.) □

4.7. *Proposition*

Let g, h be in rt-normal form and suppose $g \equiv_{\mathcal{RT}} h$. Then g, h are identical.

Proof

The proof consists of remarking that an rt-normal form g can be uniquely reconstructed from its ready trace set. The reconstruction is as follows. Let the elements of $\mathcal{RT}[\![g]\!] \cup \{o\}$ be the nodes in a process graph g^* to be constructed. The root of g^* is $I(g)$ (or $(\lambda; I(g))$ in the $(\sigma; \vec{X})$ notation). The edges of g^* are given by

$(\sigma; \vec{X}) \xrightarrow{a} (\sigma a; \vec{X}, Y)$ if the last entry of $\vec{X}$ is not $\varnothing$

$(\sigma; \vec{X}, \varnothing) \xrightarrow{\delta} o$.

Now it is a routine matter to show that g^* is in fact isomorphic to $T(g)$. So we have proved that $T(g)$ and $T(h)$ are identical (or isomorphic). From this it readily follows that g, h are identical. □

Although we will not need it, let us remark the following fact:

4.7.1. *Corollary*

Process graph transformations $\Longrightarrow\!\!\!\!\!\Longrightarrow$ are confluent.

Proof

Immediate from Propositions 4.4, 4.6 and 4.7. □

4.8. *Lemma*

$g \Longleftrightarrow\!\!\!\!\!\Longrightarrow h$ iff $g \equiv_{\mathcal{RT}} h$.

Proof

The implication $\Longrightarrow$ is immediate from Proposition 4.4. For the reverse implication, suppose g, h are rt-equivalent. Let g^* be an rt-normal form of g, obtained by a maximal $\Longrightarrow$-graph reduction (it exists by Proposition 4.6). Likewise h^* is a rt-normal form of h. Then by Proposition 4.4, $g \equiv_{\mathcal{RT}} g^*$ and $h \equiv_{\mathcal{RT}} h^*$, and hence $g^* \equiv_{\mathcal{RT}} h^*$. So by Proposition 4.7, g^* and h^* are identical, and therefore $g \Longrightarrow\!\!\!\!\!\Longrightarrow g^* \Longleftarrow\!\!\!\!\!\Longrightarrow h$.

4.9. *Lemma*

$\equiv_{\mathcal{RT}}$ is a congruence on $\mathbb{H}(+, ., \|, \between, |, \partial_H, \theta)$.

Proof

Using the previous lemma, it suffices to prove:
if $g \quad g'$, $h \quad h'$ then

(i) $g \,\square\, h \Longleftrightarrow\!\!\!\!\!\Longrightarrow g' \,\square\, h'$, where $\square$ is $+, ., \|, \between, |$.
(ii) $\partial_H(g) \Longleftrightarrow\!\!\!\!\!\Longrightarrow \partial_H(g')$
(iii) $\theta(g) \Longleftrightarrow\!\!\!\!\!\Longrightarrow \theta(g')$.

Of these implications (ii), (iii) are trivial.

(i) is easy for the operations $+$, $.$. For $\|$ it suffices to prove $g \| h \Rightarrow g' \| h$, which can be seen best using some 'geometrical intuition' (cf. Example 4.10); or, alternatively, one proves more directly that $g \Longrightarrow g'$ implies $g \| h \equiv_{\mathcal{RT}} g' \| h$. The details of a really rigorous proof would be extremely time- and space-consuming, and we will not attempt one in the present note.

The operators $\between$ and $|$ present no special difficulties. □

4.10 *Example*

See Fig. 9. Let $a|\bar{a} = a^\circ$ be the only non-trivial communication.

4.11. *Remark*

In Bergstra, Klop and Olderog[2] it was proved that readiness equivalence on $\mathbb{H}(\equiv_{\mathcal{R}})$ is generated by the graph transformations $\overset{[i]}{\Longrightarrow}$ (double edge), $\overset{[ii]}{\Longrightarrow}$ (sharing) as before, together with the transformation 'cross', by which in a part as in Fig. 10(*a*) two *b*-steps may be inserted to yield the part as in Fig. 10(*b*).

Using this fact we can pinpoint once more (cf. also Proposition 2.4) why $\equiv_{\mathcal{R}}$ is *not* a congruence w.r.t. θ. Namely, if g, g' are the graphs in Fig. 11(*a*), then $\theta(g)$, $\theta(g')$, with priorities $d > b > f$, are as in Fig. 11(*b*).

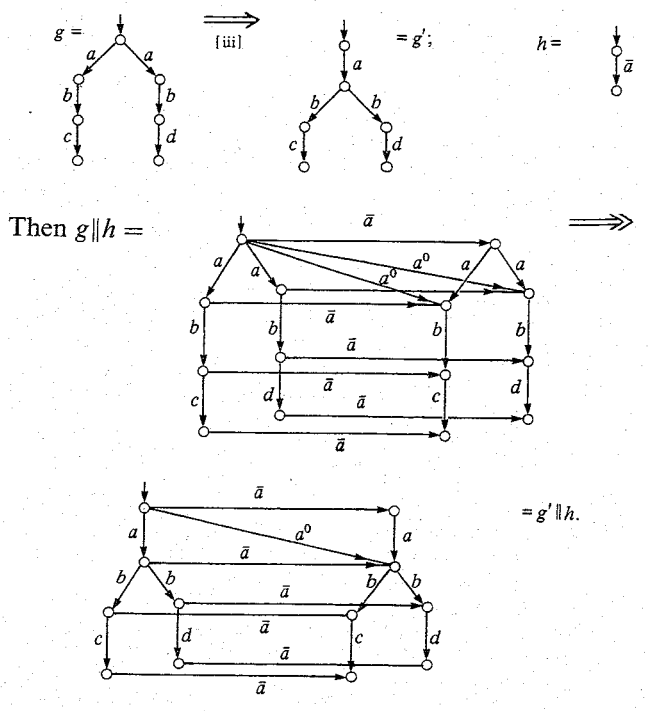

Figure 9. (Note that while $g \overset{[iii]}{\Longrightarrow} g'$, the sequence of transformations $g \| h \Longrightarrow\!\!\!\!\!\Longrightarrow g' \| h$ uses all three types of transformation.)

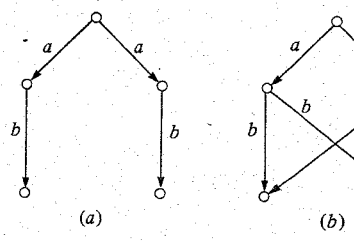

Figure 10.

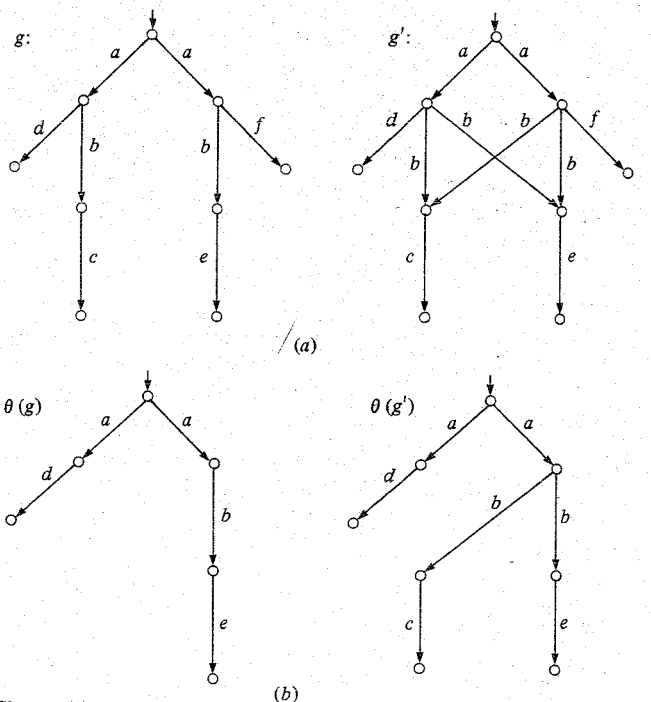

Figure 11.

Now clearly $\theta(g)$ and $\theta(g')$ are not convertible via [i], [ii] and the cross transformation. Contrast this with the present situation, where via some case analysis it is easily seen that if $g \underset{[iii]}{\Longrightarrow} g'$ then: $\theta(g) \underset{[iii]}{\Longrightarrow} \theta(g')$ or $\theta(g)$, $\theta(g')$ coincide.

4.12. *Some auxiliary operators on* $\mathbb{H}$

We will add two more operators on $\mathbb{H}$, which serve to axiomatise θ in a finite way and to formulate a proof rule typical for *RTS*.

4.12.1 *Definition*

Let $g, h \in \mathbb{H}$. Then $g \lhd h$ (g 'unless' h) is defined as the result of erasing in g all *initial* steps which are majorised (w.r.t. the p.o. $<$ to which the priority operator refers) by some *initial* step in h. (Of course, disconnected pieces are discarded.)

4.12.2. *Example*

Let $a < b < c$. Then:

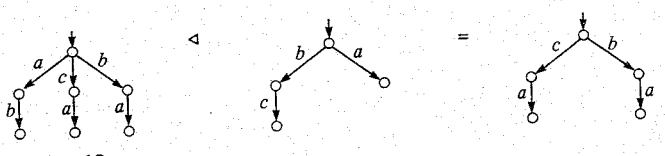

Figure 12.

4.12.3. *Definition*

$\pi_n: \mathbb{H} \longrightarrow \mathbb{H}$ is the nth projection operator ($n \geqslant 1$), which cuts off all branches after n steps. E.g.

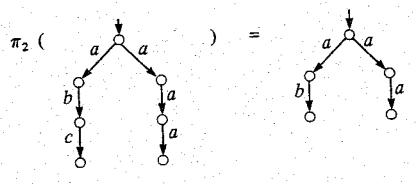

Figure 13.

It is left to the reader to check that Lemma 4.9 generalises to the presence of $\lhd$ and the $\pi_n (n \geqslant 1)$.

5. A COMPLETE AXIOMATISATION FOR READY-TRACE SEMANTICS

The previous results lead at once to a complete axiomatisation *RTS* as in Table 2 for ready-trace semantics on finite processes with communication (but without silent moves).

Here the upper part of Table 2 is *ACP*, Algebra of Communicating Processes; a, b vary over A_δ. It contains BPA_δ. The π_n are projection operators; to formulate *RTR*, the ready-trace rule, π_1 is sufficient. Note that (at least for finite processes) *RTR* is equivalent to the rule

$$\frac{\pi_1(x) = \pi_1(y)}{a(x+y) = ax+ay}$$

Table 2

RTS	$x+y = y+x$	$A1$
	$x+(y+z) = (x+y)+z$	$A2$
	$x+x = x$	$A3$
	$(x+y)z = xz+yz$	$A4$
	$(xy)z = x(yz)$	$A5$
	$x+\delta = x$	$A6$
	$\delta x = \delta$	$A7$
	$a \mid b = b \mid a$	$C1$
	$(a \mid b) \mid c = a \mid (b \mid c)$	$C2$
	$\delta \mid a = \delta$	$C3$
	$x \parallel y = x \between y + y \between x + x \mid y$	$CM1$
	$a \between x = ax$	$CM2$
	$ax \between y = a(x \parallel y)$	$CM3$
	$(x+y) \between z = x \between z + y \between z$	$CM4$
	$(ax) \mid b = (a \mid b)x$	$CM5$
	$a \mid (bx) = (a \mid b)x$	$CM6$
	$(ax) \mid (by) = (a \mid b)(x \parallel y)$	$CM7$
	$(x+y) \mid z = x \mid z + y \mid z$	$CM8$
	$x \mid (y+z) = x \mid y + x \mid z$	$CM9$
	$\partial_H(a) = a$ if $a \notin H$	$D1$
	$\partial_H(a) = \delta$ if $a \in H$	$D2$
	$\partial_H(x+y) = \partial_H(x)+\partial_H(y)$	$D3$
	$\partial_H(xy) = \partial_H(x)\partial_H(y)$	$D4$
	$\pi_m(a) = a$	$PR1$
	$\pi_1(ax) = a$	$PR2$
	$\pi_{m+1}(ax) = a\pi_m(x)$	$PR3$
	$\pi_m(x+y) = \pi_m(x)+\pi_m(y)$	$PR4$
	$\dfrac{\pi_1(x) = \pi_1(y)}{z(x+y) = zx+zy}$	RTR

5.1. *Lemma*

$BPA_\delta + PR1-4 + RTR$ is a complete axiomatisation of $\mathbb{H}(+, ., \pi_n)/\equiv_{\mathcal{RT}}$.

Proof

We refer to Bergstra, Klop and Olderog[2] for the (obvious) interpretation of process expressions in the graph model and for the arguments concerning $+$, $.$. Clearly, *RTR* corresponds to the process graph transformation $\underset{[iii]}{\Longrightarrow}$ By Lemma 4.8 we have completeness. □

The extension to the priority operator θ can easily be done on the basis of the axiomatisation ACP_θ introduced and analysed in Baeten, Bergstra and Klop[1]. ACP_θ is the axiom system consisting of *ACP* (upper part of Table 2) and the nine axioms in Table 3:

Table 3

$a \lhd b = a$ if not $(a < b)$	$P1$
$a \lhd b = \delta$ if $a < b$	$P2$
$x \lhd yz = x \lhd y$	$P3$
$x \lhd (y+z) = (x \lhd y) \lhd z$	$P4$
$xy \lhd z = (x \lhd z)y$	$P5$
$(x+y) \lhd z = x \lhd z + y \lhd z$	$P6$
$\theta(a) = a$	$TH1$
$\theta(xy) = \theta(x)\theta(y)$	$TH2$
$\theta(x+y) = \theta(x) \lhd y + \theta(y) \lhd x$	$TH3$

We will refer to *RTS* together with the axioms in Table 3 as: RTS_θ.

5.2. *Theorem*

RTS_θ is a complete axiomatisation of the graph model

$$\mathbb{H}(+, ., \|, \lfloor\!\lfloor, |, \partial_H, \pi_n, \lhd, \theta)/\equiv_{\mathscr{RT}}$$

Proof

The proof is entirely similar to the one for readiness semantics in Bergstra, Klop and Olderog[2], using Lemma 5.1 and the fact that all operators except the basic operators $+$, . can be eliminated from process expressions. □

5.3. *Corollary*

RTS_θ is a consistent (in the sense of Section 2.2) axiomatisation.

Proof

The elements of $\mathbb{H}(-)/\equiv_{\mathscr{RT}}$ are $\equiv_{\mathscr{RT}}$-equivalence classes of graphs; each equivalence class is generated by the three graph transformations of Section 3. These transformations preserve traces. Hence, by Lemma 5.2, the axiom system RTS_θ is consistent. □

6. AN EXPLICIT PRESENTATION OF THE MODEL FOR READY-TRACE SEMANTICS

Above, we have obtained a model for *RTS*, with equivalence classes of process graphs as elements. It is also possible to present this model in a more direct way, namely with the ready-trace sets themselves as elements (without any mention of underlying process graphs). This requires formulating some closure properties of ready-trace sets (Cf. the analogous procedure in Bergstra, Klop and Olderog[2] for failure semantics.) The end result will be an 'explicit' model for *RTS*, which is isomorphic to the 'graph model' above. In order to obtain this explicit representation, we have to define the operations $+$, ., $\|$, $\lfloor\!\lfloor$, $|$, ∂_H, θ, $\lhd$, π_n directly on the ready-trace sets. We will do this only for θ, and further be satisfied with the formulation of the closure properties inherent in a ready-trace set.

6.1. *Definition*

(i) Let $X_0, a_0, X_1, a_1, \ldots, a_{n-1}, X_n$ be an alternating sequence of sets X_i $(i = 0, \ldots, n)$ and actions a_i $(i = 0, \ldots, n-1)$ for some $n \geqslant 0$. Then this sequence is a *ready trace* if

(1) $a_i \in X_i$ $(i < n)$

(2) $X_0, \ldots, X_{n-1} \subseteq A$

(3) $X_n \subseteq A$ or $X_n = \{\varepsilon\}$ $(\varepsilon \notin A)$.

We recall the notation $(\sigma; \vec{X}) = (a_0, \ldots, a_{n-1}; X_0, \ldots, X_n)$ for $X_0, a_0, \ldots, X_n$.

(ii) Let $\mathscr{X}$ be a collection of ready traces. Then $\mathscr{X}$ is a *ready trace set* if $\mathscr{X}$ satisfies the following clauses:

(1) if $(\sigma; \vec{X}) \in \mathscr{X}$ and $(\rho; \vec{Y}) \in \mathscr{X}$, then $X_0 = Y_0$ (*root condition*)

(2) if $(\sigma; \vec{X})$ is a ready trace and $(\sigma\rho; \vec{X}\,\vec{Y}) \in \mathscr{X}$ then $(\sigma; \vec{X}) \in \mathscr{X}$ (*prefix condition*)

(3) if $(a_0, \ldots, a_{k-1}; X_0, \ldots, X_k) \in \mathscr{X}$ and $a_k \in X_k$ $(a_k \neq \varepsilon)$, then $(a_0, \ldots, a_{k-1}, a_k; X_0, \ldots, X_k, X_{k+1}) \in \mathscr{X}$ for some X_{k+1} $(\subseteq A$ or $= \{\varepsilon\})$ (*continuation condition*).

Clearly any $\mathscr{RT}[\![g]\!]$, $g \in \mathbb{H}$ is a ready-trace set in this definition. (It holds moreover for $\mathscr{RT}[\![g]\!]$, g being in this paper finite and acyclic, that every ready trace in $\mathscr{RT}[\![g]\!]$ can be continued to one in which the last ready set is $\varnothing$ or $\{\varepsilon\}$.)

Vice versa, we can associate a process graph $g_{\mathscr{X}}$ (in fact a tree) to a ready-trace set $\mathscr{X}$ as already explained in the proof of Proposition 4.7.

An alternative definition of a ready-trace set would be one in which the ready traces are allowed to be infinite. Under that definition, the resulting semantics would distinguish processes like $\sum_n a^n$ and $\sum_n a^n + a^\omega$.

It is possible to define the operators considered above directly on these ready-trace sets. We will not do that, except for the case of θ, in order to understand better why θ is compatible with RTS – and not with the coarser semantics such as readiness or failure semantics.

6.2. *Definition*

Let a p.o. $<$ on A_δ be given.

(i) Let $X \subseteq A$. Then $\theta(X)$ is the set of *maximal* elements (w.r.t. $<$) in X. If $X = \{\varepsilon\}$, $\theta(X) = X$. If $\vec{X} = X_0, \ldots, X_n$, $\theta(\vec{X}) = \theta(X_0), \ldots, \theta(X_n)$.

(ii) Let $(\sigma; \vec{X})$ be a ready trace. Then $(\sigma; \theta(\vec{X}))$ need not be a ready trace, since property (1) of Definition 5.4.1 may be violated. However, $(\sigma; \theta(\vec{X}))$ contains a maximal prefix (in the obvious sense) which is still a ready trace. Now we define

$$\theta(\sigma; \vec{X})$$

to be this maximal prefix ready trace of $(\sigma; \theta(\vec{X}))$.

(*Example*: if $a < b$ then $\theta(\{a, b\}, b, \{a, b\}, a, \varnothing) = \{b\}, b, \{b\}$.)

(iii) Let $\mathscr{X}$ be a ready-trace set. Then

$$\theta(\mathscr{X}) = \{\theta(\sigma; \vec{X}) \mid (\sigma; \vec{X}) \in \mathscr{X}\}.$$

Now we claim (without proof) that θ and $\mathscr{RT}[\![\,]\!]$ commute:

6.3. *Claim*

Let $g \in \mathbb{H}$. Then:

$$\mathscr{RT}[\![\theta(g)]\!] = \theta(\mathscr{RT}[\![g]\!]).$$

Note that in this explicit definition of the operator θ on ready-trace sets it is essential to have the intermediate ready sets in a ready trace available.

7. CONCLUDING REMARKS

We have considered a process semantics *RTS*, ready-trace semantics (called in Pnueli[8] 'barbed semantics'), which is intermediate between bi-simulation semantics (*BS*) and readiness semantics (*RS*). The advantage of *RTS* over *RS* and *FS* (failure semantics) is that it allows the presence of operators that may be important for process specification, such as the priority operator θ.

That is in contrast with *RS* and *FS*, which reject operators like θ, since in these semantics too many

processes are identified to bear the presence of θ. This seems to be a general phenomenon: the finer the process equivalence, the more operators on processes (like θ) can be defined. Adding more equations, i.e. making the process equivalence coarser, increases the ease of process verifications, but at the cost of losing specification possibilities by means of operators such as θ, which become undefinable.

From an intuitive point of view (see Remark 3.9) a semantics such as *RTS* seems perfectly natural.

An evident direction for further work is the extension of the above to infinite processes, and to silent moves.*

* R. van Glabbeek has informed us that there is a neat complete axiomatisation of *RTS* for finite processes with τ-steps without, however, the priority operator θ.

REFERENCES

1. J. C. M. Baeten, J. A. Bergstra and J. W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fund. Inf. IX (2), 1986, pp. 127–168.
2. J. A. Bergstra, J. W. Klop and E.-R. Olderog, *Readies and Failures in the Algebra of Communicating Processes*. CWI report CS-R8523, Centre for Mathematics and Computer Science, Amsterdam (1985).
3. J. A. Bergstra, J. W. Klop and E.-R. Olderog, *Failure Semantics with Fair Abstraction*. CWI report CS-R8609, Amsterdam (1986).
4. S. D. Brookes, On the relationship of CCS and CSP. In *Proc. 10th ICALP, Barcelona*, edited J. Díaz. Springer LNCS no. 154, pp. 83–96 (1983).
5. S. Brookes, C. Hoare and W. Roscoe, A theory of communicating sequential processes. *Journal of the Computing Association of Machines* **31** (3), 560–599 (1984).
6. R. Milner, *A Calculus of Communicating Systems*. Springer LNCS no. 92 (1980).
7. E.-R. Olderog and C. A. R. Hoare, Specification-oriented semantics for communicating processes. In *Proc. 10th ICALP, Barcelona*, edited J. Díaz. Springer LNCS no. 154 (1983). Expanded version: Technical Monograph PRG-37, Oxford University Computer Laboratory (1984).
8. A. Pnueli, Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion*, edited W. Brauer. Springer LNCS no. 194, pp. 15–32 (1985).