Technical Report

UCAM-CL-TR-157
ISSN 1476-2986

Number 157

UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

Introducing a priority operator to CCS

Juanito Camilleri

January 1989

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Introducing a Priority Operator to CCS

Juanito Camilleri
University of Cambridge,
Computer Laboratory, New Museum Site,
Pembroke Street,
Cambridge CB2 3QG.

January 9, 1989

Abstract

In this paper we augment the syntax of CCS by introducing a priority operator. We present a syntax directed operational semantics of the language as a labelled transition system. A new equivalence relation which is based on Milner's strong observational equivalence [11] is defined and proved to be a congruence. We also give some examples which illustrate the use of the operator and emphasise the novelty of the approach used to introduce the notion of priority to process algebras.

1 Introduction

In retrospect, we see many attempts to define the semantics of communicating processes with the hope that this would yield theories that help us gain a better understanding of the behaviour of such processes. Many semantic theories that have been developed with this aim [e.g., CCS, CSP, SCCS] are operational in nature; the semantics of the languages underlying these theories can be defined using labelled transition systems [13]. In most existing theories of communicating processes, it is assumed that all actions performed by a process, have equal importance. This is clearly not satisfactory because there are many applications [e.g., the handling of interrupts] in which some actions performed by a process have a greater importance than others.

There have been a few attempts at defining an operational semantics for an algebraic theory of concurrency that includes a notion of priority [1] [5]. Not withstanding this, it is not clear which approach is the most appealing and generally applicable. In this paper we propose another attempt at including priority in a process algebra. The approach suggested by Baeten et al., [1], does not include an operational semantics or a behavioural equivalence that underlies the theory that is presented. Our approach is closer in style to that adopted by Cleaveland

and Hennessy [5]. Nonetheless, it differs in the fact that we give a syntax directed operational semantics to a priority operator $+\!\!\!\!/$ which is used to define certain situations in which an action has a higher priority than another action. Therefore we do not preassign priorities to actions by a partial ordering on the alphabet of actions. The ideas adopted in this paper have evolved from work done on defining the operational semantics of priority alternation [4] which is one of the constructs of occam [10].

The remainder of this paper has been divided into the following sections. In section 2 we introduce the language to be used and define its operational semantics. In section 3 we give some examples to illustrate the operation of the priority operator in certain situations. In Section 4 we compare our method of introducing the notion of priority to a process algebra, to other approaches presented in the literature. Section 5 deals with a new behavioural equivalence based on strong observational equivalence [11]. We prove that this equivalence relation is in fact a congruence and in Section 6, we state some results. Finally in section 7 we discuss the work and outline our conclusions.

2 The Language

The syntax of our language is basically that of pure CCS with the addition of a priority operator $+\!\!\!\!/$. Let Δ be the set of names ranged over by α,β,γ. We use $\overline{\Delta} = \{\overline{\alpha} \mid \alpha \in \Delta\}$ to denote the set of conames which is disjoint from Δ and is in bijection with it, under the map $\alpha \mapsto \overline{\alpha}$. Let $\Lambda = \Delta \cup \overline{\Delta}$ be the set of labels ranged over by λ then $Act = \Lambda \cup \{\tau\}$ is the set of actions ranged over by μ. We use θ to denote a relabelling; a mapping from Act to Act such that $\theta(\tau) = \tau$. We define our language of terms as follows. Let $t \in$ **Terms**, then

$$t ::= nil \quad | \quad \mu.t \quad | \quad t|t \quad | \quad t \setminus \lambda \quad | \quad t[\theta] \quad | \quad t + t \quad | \quad t +\!\!\!\!/ t \quad | \quad fix(x.t)$$

Before attempting to formalise the semantics of the $+\!\!\!\!/$ operator let us try to build an intuitive understanding of its operation. Consider the construct $t \equiv \alpha.t_0 +\!\!\!\!/ \beta.t_1$. The construct t describes a situation where an α-action is required to have a higher priority than a β-action. Therefore t can perform an α-action and become t_0 without any constraints, but, t can only perform a β-action to become t_1 provided the parallel context of t isn't willing to perform an $\overline{\alpha}$-action. In other words, if t is executing in parallel with t' (i.e., t' is the parallel context of t) then t can perform a β-action to become t_1 provided t' refuses to perform an $\overline{\alpha}$-action. On the other hand if t' can perform an $\overline{\alpha}$-action then this guarantees that t will not perform a β-action.

It is evident that we have to formalise the notion of a term refusing to perform some actions and accepting to perform others. In the following sections we define the refusal and acceptance sets.

2.1 The Refusal Set

Informally, a refusal set of a term t can be described as some set of actions which does not include any action that can take place next in t. For example, if $t \equiv \alpha.t_0 + \beta.t_1$ then t refuses any set R provided $\alpha, \beta \notin R$. Let $\mathbf{ref} \subseteq \text{Terms} \times \mathcal{P}_{fin}(Act)$ be defined by rules $(i)^a$-$(viii)^a$. If t **ref** R holds we say t refuses R and we call R a refusal set of t. (Note $\mathcal{P}_{fin}(Act)$ denotes the finite powerset of actions Act).

$(i)^a$ nil **ref** R

$(ii)^a$ $\mu.t$ **ref** R *provided* $\mu \notin R$.

$(iii)^a$

$$\frac{t \text{ ref } R - \{\lambda, \overline{\lambda}\}}{t \setminus \lambda \text{ ref } R}$$

$(iv)^a$

$$\frac{t \text{ ref } R}{t[\theta] \text{ ref } R[\theta]}$$

$(v)^a$

$$\frac{t_0 \text{ ref } R \quad t_1 \text{ ref } R}{t_0 + t_1 \text{ ref } R}$$

$(vi)^a$

$$\frac{t_0 \text{ ref } R \quad t_1 \text{ ref } R}{t_0 \Rightarrow t_1 \text{ ref } R}$$

$(vii)^a$

$$\frac{t_0 \text{ ref } R \quad t_1 \text{ ref } R}{t_0 | t_1 \text{ ref } R}$$

$(viii)^a$

$$\frac{t[fix(x.t)/x] \text{ ref } R}{fix(x.t) \text{ ref } R}$$

Lemma 1 If p **ref** R, then for $R' \subseteq R$, p **ref** R'.

Proof

We proceed by induction on the structure of derivations that for all R, R', if p **ref** R and $R' \subseteq R$ then this implies that p **ref** R'. We proceed by performing a case analysis on p.

1. When $p \equiv nil$ or $p \equiv \tau.$t the proof is trivial.

2. $\underline{p \equiv \lambda.t}$

   By rule $(ii)^a$, $\lambda.t$ **ref** R for any R provided $\lambda \notin R$. Since $R' \subseteq R$ then $\lambda \notin R'$. Hence $\lambda.t$ **ref** R'.

3. $\underline{p \equiv t \setminus \lambda}$

   Suppose $t \setminus \lambda$ **ref** R, then by shorter inference, t **ref** $R - \{\lambda, \overline{\lambda}\}$. Now $R' \subseteq R$, therefore $R' - \{\lambda, \overline{\lambda}\} \subseteq R - \{\lambda, \overline{\lambda}\}$. Hence $t \setminus \lambda$ **ref** R'. A similar argument can be used when $p \equiv t[\theta]$.

4. $\underline{p \equiv t_0 \leftthreetimes t_1}$

   Let $t_0 \leftthreetimes t_1$ **ref** R, then by shorter inference one can say that t_0 **ref** R and t_1 **ref** R. Applying the induction hypothesis on t_0 and t_1 yields t_0 **ref** R', t_1 **ref** R'. Therefore by rule $(vi)^a$, $t_0 \leftthreetimes t_1$ **ref** R'. A similar argument can be used for $p \equiv t_0 + t_1$ and $p \equiv t_0 | t_1$.

5. $\underline{p \equiv fix(x.t)}$

   Let $fix(x.t)$ **ref** R, then by shorter inference $t[fix(x.t)/x]$ **ref** R. Therefore $t[fix(x.t)/x]$ **ref** R'. Hence by rule $(viii)^a$ $fix(x.t)$ **ref** R'.

□

2.2 The Acceptance Set

Informally, the acceptance set of a term t can be described as the set of complements of actions which can take place next in t. For example if $t \equiv \alpha.t_0 + \beta.t_1$, then the acceptance set of t should be $\{\overline{\alpha}, \overline{\beta}\}$. Let $*$ be a special symbol corresponding to idling which is viewed as a complement to τ. We use the notation $\overline{\tau} \equiv *$. Therefore the acceptance set of $\tau.t_0 + \beta.t_1$ which should be $\{*, \overline{\beta}\}$, is denoted by $\{\overline{\tau}, \overline{\beta}\}$.

Let **acc** $\subseteq$ Terms $\times \mathcal{P}_{fin}(Act \cup \{*\})$ be defined by rules $(i)^b - (viii)^b$ below. If t **acc** A holds then we say t accepts A and we call A the acceptance set of t.

$(i)^b$ $\quad nil$ **acc** $\emptyset$.

$(ii)^b$ $\quad \mu.t$ **acc** $\{\overline{\mu}\}$

$(iii)^b$ $\quad \dfrac{t \text{ acc } A}{t \setminus \lambda \text{ acc } A - \{\lambda, \overline{\lambda}\}}$

$(iv)^b$ $\quad \dfrac{t \text{ acc } A}{t[\theta] \text{ acc } A[\theta]}$

$(v)^b$ $\quad \dfrac{t_0 \text{ acc } A_0 \quad t_1 \text{ acc } A_1}{t_0 + t_1 \text{ acc } A_0 \cup A_1}$

$(vi)^b$ $\quad \dfrac{t_0 \text{ acc } A_0 \quad t_1 \text{ acc } A_1}{t_0 \leftthreetimes t_1 \text{ acc } A_0 \cup A_1}$

$(vii)^b$ $\quad \dfrac{t_0 \text{ acc } A_0 \quad t_1 \text{ acc } A_1}{t_0 | t_1 \text{ acc } A_0 \cup A_1}$

$(viii)^b$ $\quad \dfrac{t[fix(x.t)/x] \text{ acc } A}{fix(x.t) \text{ acc } A}$

2.3 The Semantics of the Language

Processes are interpreted using labelled transition systems. These are triples of the form $\langle P, \longrightarrow, Act\rangle$ where P is the set of closed terms, Act is the set of actions and $\longrightarrow \subseteq \mathcal{P}_{fin}(Act) \times P \times Act \times P$ is a relation defining the behaviour of processes as shown by rules $(i)^c - (xii)^c$. Let $\vdash_R t_0 \xrightarrow{\mu} t_0'$ denote t_0 performs μ to yield t_0' provided the following condition is satisfied; if there is any process t_1 executing in parallel with t_0 then t_1 must refuse R.

$(i)^c \quad \vdash_R \mu.t \xrightarrow{\mu} t$

$(ii)^c$

$$\frac{\vdash_R t \xrightarrow{\mu} t'}{\vdash_R t \setminus \lambda \xrightarrow{\mu} t' \setminus \lambda} \quad \text{if } \mu \neq \lambda, \overline{\lambda}$$

$(iii)^c$

$$\frac{\vdash_R t \xrightarrow{\mu} t'}{\vdash_R t[\theta] \xrightarrow{\theta(\mu)} t'[\theta]}$$

$(iv)^c$

$$\frac{\vdash_R t_0 \xrightarrow{\mu} t_0' \quad t_1 \text{ ref } R}{\vdash_R t_0|t_1 \xrightarrow{\mu} t_0'|t_1}$$

$(v)^c$

$$\frac{\vdash_R t_1 \xrightarrow{\mu} t_1' \quad t_0 \text{ ref } R}{\vdash_R t_0|t_1 \xrightarrow{\mu} t_0|t_1'}$$

$(vi)^c$

$$\frac{\vdash_{R_0} t_0 \xrightarrow{\mu} t_0' \quad t_0 \text{ ref } R_1 \qquad \vdash_{R_1} t_1 \xrightarrow{\overline{\mu}} t_1' \quad t_1 \text{ ref } R_0}{\vdash_{R_0 \cup R_1} t_0|t_1 \xrightarrow{\tau} t_0'|t_1'}$$

$(vii)^c$

$$\frac{\vdash_R t_0 \xrightarrow{\mu} t_0'}{\vdash_R t_0 + t_1 \xrightarrow{\mu} t_0'}$$

$(viii)^c$

$$\frac{\vdash_R t_1 \xrightarrow{\mu} t_1'}{\vdash_R t_0 + t_1 \xrightarrow{\mu} t_1'}$$

$(ix)^c$

$$\frac{\vdash_R t_0 \xrightarrow{\mu} t_0'}{\vdash_R t_0 \Rightarrow t_1 \xrightarrow{\mu} t_0'}$$

$(x)^c$

$$\frac{\vdash_R t_1 \xrightarrow{\mu} t_1' \quad t_0 \text{ acc } A}{\vdash_R t_0 \Rightarrow t_1 \xrightarrow{\mu} t_1'} \quad \text{if } A \subseteq R$$

$(xi)^c$

$$\frac{\vdash_R t[fix(x.t)/x] \xrightarrow{\mu} t'}{\vdash_R fix(x.t) \xrightarrow{\mu} t'}$$

$(xii)^{c\,1}$

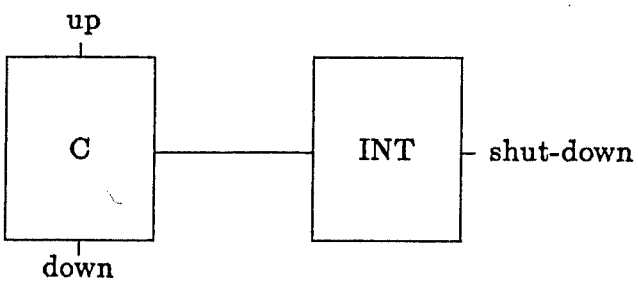
$$\frac{\vdash_R t \xrightarrow{\mu} t'}{\vdash_S t \xrightarrow{\mu} t'} \quad \text{if } R \subseteq S$$

[1] Rule $(xii)^c$ is required in the proof of Part (ii) of Lemma 2 and Case 4 of Lemma 3.1.

3 Examples

3.1 Example 1

Figure 1



The aim of this first example is to justify the need for the notion of priority in a process algebra such as CCS. The system illustrated in *Figure* 1 [5], describes a component C which acts as a counter, while INT is designed to halt C as soon as the environment issues a *shut-down* request. In the following description of the behaviour of the above system, we use some standard syntactic sugar which can be translated easily to our original language of Terms.

$$
\begin{array}{lcl}
SYS & \Leftarrow & (C_n \mid INT) \setminus i \\
INT & \Leftarrow & \textit{shut-down}.\bar{i}.nil \\
C_0 & \Leftarrow & up.C_1 + i.nil \\
C_{n+1} & \Leftarrow & up.C_{n+2} + down.C_n + i.nil \\
C_{lim} & \Leftarrow & down.C_{lim-1} + i.nil
\end{array}
$$

The system described above can perform a series of *up* or *down* actions. It can only perform a finite number of consecutive *up* actions which is determined by the value assigned to *lim* (Similarly for *down* actions). Let us assume that *lim* is a finitely large number. It is evident that for $0 < n \leq lim$, *down up shut-down down up* ... is a sequence of actions which may take place in SYS. In this case although the action *shut-down* is performed, C_n may choose to ignore indefinitely the offer of communication from INT along channel i. Since we require that SYS halts as soon as the environment issues a *shut-down* request, we conclude that the above definition is not satisfactory. The situation described above is typical of interrupts, time-outs in communication protocols and various other real time applications where it is vital that in certain situations some actions take priority over others thus avoiding the risk of being ignored indefinitely. The + operator is therefore unsatisfactory to describe real time applications such as the one above because an urgent action might be ignored until it is too late and perhaps indefinitely. We have introduced a new operator $+\!\!\!+$ which can be used to solve our problem as follows:

$$
\begin{array}{lcl}
SYS & \Longleftarrow & (C_n \mid INT) \setminus i \\
INT & \Longleftarrow & shut\text{-}down.\overline{i}.nil \\
C_0 & \Longleftarrow & i.nil \nplus up.C_1 \\
C_{n+1} & \Longleftarrow & i.nil \nplus (up.C_{n+2} + down.C_n) \\
C_{lim} & \Longleftarrow & i.nil \nplus down.C_{lim-1}
\end{array}
$$

The semantics of the $\nplus$ operator is defined to ensure that actions on the right hand side of the operator can only take place provided none of the actions on the left hand side can take place. Therefore C_n can only perform an *up* or *down* action provided the environment hasn't issued a *shut-down* request. Once a *shut-down* request is issued, no more *up* or *down* actions are possible in C_n.

Proposition 1

In the construct $C \equiv (C_2 \mid \text{INT}) \setminus i$, neither an *up* action nor a *down* action can take place once a *shut-down* action has been issued.

Outline of Proof

There is a proof tree to support any transition that can take place in a transition system. By definition $C_2 \equiv i.nil \nplus (up.C_3 + down.C_1)$. We can build a proof tree for $\vdash_R (C_2|INT) \setminus i \stackrel{shut-down}{\longrightarrow} (C_2|\overline{i}.nil) \setminus i$ using rules $(ii)^c$, $(iv)^c$, $(ix)^c$, $(i)^c$ and $(ii)^a$. We proceed to prove proposition 1 by arguing that a proof tree cannot be constructed for an *up* or *down* action as the next transtion in the execution of the construct $(C_2|\overline{i}.nil) \setminus i$.

Consider $((i.nil \nplus (up.C_3 + down.C_1))|\overline{i}.nil) \setminus i$. According to rule $(x)^c$ an *up* action or a *down* action can take place provided $\{\overline{i}\} = A \subseteq R$ where $i.nil$ acc A. By definition $\overline{i}.nil$ ref R provided $\overline{i} \notin R$. Therefore we have a contradiction; rule $(x)^c$ is only possible provided $\{\overline{i}\} \subseteq R$, but, rule $(iv)^c - (vi)^c$ can only be matched provided $\overline{i}.nil$ ref R, which in turn requires $\overline{i} \notin R$. Hence we cannot construct proof trees for *up* or *down* actions from $(C_2|i.nil) \setminus i$. In other words, once *shut-down* is issued the only possible action which can be justified by a proof tree is a communication via channel i. The proof tree of this transition illustrated in Figure 2 was constructed using rules $(ii)^c$, $(vi)^c$, $(ix)^c$ together with rules $(vi)^a$, $(v)^a$ and $(ii)^a$.

□

It is interesting that Proposition 1 does not hold if one considers $INT \Longleftarrow shut\text{-}down.\tau.\overline{i}.nil$ in an interleaving semantics which need not be fair. In this case, an infinite number of *up* and *down* actions may occur after a *shut-down* request. Therefore τ never occurs and $\overline{i}$ is never enabled. In order to solve this problem one can adopt the ideas on fairness presented by Costa and Stirling [6].

Figure 2

$$\frac{\vdash_{R_0} i.nil \xrightarrow{i} nil \qquad up, down, i \notin R_1 \qquad \bar{i} \notin R_0}{\vdash_{R_0} (i.nil \leftthreetimes (up.C_3 + down.C_1)) \xrightarrow{i} nil \qquad C_2 \text{ ref } R_1 \qquad \vdash_{R_1} \bar{i}.nil \xrightarrow{\bar{i}} nil \qquad \bar{i}.nil \text{ ref } R_0}$$

$$\frac{\vdash_{R_0} (i.nil \leftthreetimes (up.C_3 + down.C_1)) \xrightarrow{i} nil \qquad C_2 \text{ ref } R_1 \qquad \vdash_{R_1} \bar{i}.nil \xrightarrow{\bar{i}} nil \qquad \bar{i}.nil \text{ ref } R_0}{\vdash_{R_0 \cup R_1} (i.nil \leftthreetimes (up.C_3 + down.C_1)) \mid \bar{i}.nil \xrightarrow{\tau} nil}$$

$$\frac{\vdash_{R_0 \cup R_1} (i.nil \leftthreetimes (up.C_3 + down.C_1)) \mid \bar{i}.nil \xrightarrow{\tau} nil}{\vdash_{R_0 \cup R_1} (C_2 \mid \bar{i}.nil) \setminus i \xrightarrow{\tau} nil}$$

3.2 Example 2

The aim of this example is to emphasise the care needed when using the $\leftthreetimes$ operator. Since our approach is syntax directed, the onus is on the person writing an expression to ensure that no conflict occurs in the priority of actions. Consider the construct $C \equiv ((\alpha.\gamma.nil \leftthreetimes \beta.\phi.nil) | (\overline{\beta}.\delta.nil \leftthreetimes \overline{\alpha}.\eta.nil)) \setminus \alpha \setminus \beta$. This construct describes a situation where an α-action has a higher priority than a β-action and a $\overline{\beta}$-action has a higher priority than an $\overline{\alpha}$-action. The situation described by C is an example of an erroneous definition.

Proposition 2: Construct C deadlocks.

Proof

Let us denote $(\alpha.\gamma.nil \leftthreetimes \beta.\phi.nil)$ by C_0 and $(\overline{\beta}.\delta.nil \leftthreetimes \overline{\alpha}.\eta.nil)$ by C_1. Suppose $\vdash_R C_0 | C_1 \xrightarrow{\tau} (\gamma.nil) | (\eta.nil)$, then according to rule $(vi)^c$ there exists an R_0, R_1 such that $R_0 \cup R_1 = \mathrm{R}$ and $\vdash_{R_0} C_0 \xrightarrow{\alpha} (\gamma.nil)$, $\vdash_{R_1} C_1 \xrightarrow{\overline{\alpha}} (\eta.nil)$, C_0 ref R_1, C_1 ref R_0. Now $\vdash_{R_1} C_1 \xrightarrow{\overline{\alpha}} (\eta.nil)$ is true provided $\vdash_{R_1} \overline{\alpha}.\eta.nil \xrightarrow{\overline{\alpha}} (\eta.nil)$ and $\overline{\beta}.\delta.nil$ acc $\{\beta\}$ such that $\{\beta\} \subseteq R_1$. On the other hand $(\alpha.\gamma.nil \leftthreetimes \beta.\phi.nil)$ ref R_1 and therefore by shorter inference $(\alpha.\gamma.nil)$ ref R_1, $(\beta.\phi.nil)$ ref R_1. Now by rule $(ii)^a$, $(\alpha.\gamma.nil)$ ref R_1 for any R_1 provided $\alpha \notin R_1$ and $(\beta.\phi.nil)$ ref R_1 for any R_1 provided $\beta \notin R_1$. Therefore $(\alpha.\gamma.nil \leftthreetimes \beta.\phi.nil)$ ref R_1 provided $\alpha, \beta \notin R_1$. Hence we have a contradiction, one condition requires $\{\beta\} \subseteq R_1$ (i.e. $\beta \in R_1$) while the other requires that $\beta \notin R_1$. This implies that we cannot construct a proof tree for $\vdash_R C_0 | C_1 \xrightarrow{\tau} (\gamma.nil) | (\eta.nil)$ and therefore conclude that this transition cannot take place. We can follow a similar argument for $\vdash_R C_0 | C_1 \xrightarrow{\tau} (\phi.nil) | (\delta.nil)$. Therefore we can conclude that there isn't any action that can take place in C. Hence we have deadlock.

□

4 Putting Things into Perspective

There have been other attempts to introduce the notion of priority to process algebras. In this section we present a resume of two attempts that have influenced the development of the method outlined in this paper.

J.C.M.Baeten et al. [1], introduced a priority operator θ to ACP (The Algebra of Communicating Processes). They define θ in an axiomatic way by assigning priorities to actions and by defining the operator θ in such a way as to model the preassigned priorities as explained hereafter. In order to define the θ operator they define an auxiliary operator $\lhd$: P $\times$ P $\rightarrow$ P where P ranges over the set of processes.

$x \lhd y$ is pronounced x unless y where

P1 $a \lhd b = a$ if not $(a < b)$

P2 $a \lhd b = \delta$ if $a < b$

Note δ denotes deadlock.

So $a \lhd b$ is equal to a unless b has a higher priority over a.

For example, if $a < c$ and $b < c$ then $(ax + by + c) \lhd c = \delta + \delta + c = c$.

They go on to augment ACP with rules **P1-P6** and **TH1-TH3** thus defining algebra ACP_θ.

P3 x $\lhd$ yz = x $\lhd$ y

P4 $x \lhd (y + z) = (x \lhd y) \lhd z$

P5 $xy \lhd z = (x \lhd z)y$

P6 $(x + y) \lhd z = x \lhd z + y \lhd z$

TH1 $\theta(a) = a$

TH2 $\theta(xy) = \theta(x).\theta(y)$

TH3 $\theta(x + y) = \theta(x) \lhd y + \theta(y) \lhd x$

The following example illustrates how the above rules in ACP_θ capture the notion of priority. Suppose one preassigns the ordering b<a on actions a, b, then

$$\theta(a + b) = \theta(a) \lhd b \ + \ \theta(b) \lhd a = a \lhd b \ + \ b \lhd a = a + \delta = a$$

Intuitively, in a context where b has a lower priority then a, $\theta(a + b) = a$ (i.e., a takes precedence over b).

Cleaveland and Hennessy [5], define an operational semantics for an algebraic theory of concurrency which incorporates priority in the definition of the execution of actions. For simplicity they assume a two level hierarchy of priorities; an action is either prioritized or unprioritized. If A is the set of actions then $\underline{A}$ is the set of prioritized actions such that $\underline{A} = \{\underline{\alpha} \mid \alpha \in A\}$. They define the operational semantics of priority in CCS in two stages. In the first stage they present an *a priori* semantics; i.e., the operational semantics of CCS. In the second stage they define the relations $\longrightarrow\!\!\triangleright$ representing the actions which are actually possible when introducing the notion of priority. These relations are defined by:

1. if $p \xrightarrow{\alpha} q$ then $p \xrightarrow{\underline{\alpha}}\!\!\triangleright\ q$

2. if $p \xrightarrow{\alpha} q$ and for no q', $\underline{\beta}$ does $p \xrightarrow{\underline{\beta}} q'$ then p $\xrightarrow{\alpha}\!\!\triangleright\ q$.

Intuitively, prioritized actions are unconstrained while unprioritized actions can only happen if unprioritized actions are possible. Cleaveland and Hennessy go on to apply the definition of a bisimulation [12] to the transition system $\langle P, \longrightarrow\!\!\triangleright, Act\rangle$ thus obtaining a new equivalence $\sim_p$. Unfortunately $\sim_p$ is not a congruence. Therefore they define another transition system based on a new arrow $\succ\!\!\longrightarrow\!\!\triangleright$.

They say that p is patient if $p \xrightarrow{\tau} q$ for no q, and then define $\succ\!\!\longrightarrow\!\!\triangleright$ by:

1. if $p \xrightarrow{\underline{\alpha}} q$ then $p \succ\!\!\xrightarrow{\underline{\alpha}}\!\!\triangleright\ q$

2. if $p \xrightarrow{\alpha} q$ and p is patient, then $p \succ\!\!\xrightarrow{\alpha}\!\!\triangleright\ q$.

As before prioritized actions are not constrained. However unprioritized actions are prempted by $\underline{\tau}$; i.e., they can only take place provided a communication cannot occur as a result of a handshaking between two complementary prioritized actions. The intuition as described above, is very closely related to that adopted in this report where we only allow a low priority action to take place provided at that point in the execution sequence, the parallel context is not ready to handshake with an action which has a higher priority.

Our approach differs from the ones described above in the fact that we give a syntax directed operational semantics to a priority operator $\dashv\!\!\vdash$ which is used to define certain situations in which an action has a higher priority to another action. Therefore we do not have to preassign priorities to actions by a partial ordering on the alphabet of actions (as in [1] [5]), nor do we have to introduce operators to explicitly prioritize and deprioritize actions (as in [5]).

5 The Behavioural Equivalence

So far we have defined a labelled transition system $\langle P, \longrightarrow, Act \rangle$ where P is the set of closed terms.

Definition 1: Given the labelled transition system $\langle P, \longrightarrow, Act \rangle$, a bisimulation $S \subseteq P \times P$ is a symmetric relation satisfying $\langle p, q \rangle \in S$ and for all $R, \vdash_R p \xrightarrow{\mu} p'$ implies that there exists some q' such that $\vdash_R q \xrightarrow{\mu} q'$ and $\langle p', q' \rangle \in S$.

One can use similar arguments as shown by Milner [12], to conclude that the largest such bisimulation (denoted by $\sim_p$) exists and is in fact an equivalence relation. We would like to prove a theorem that states that $\sim_p$ is a congruence. (i.e. p $\sim_p$ q implies that C[p] $\sim_p$ C[q] for all contexts C). Before attempting to prove this, we need to sort out the proofs of some lemmas whose results are required in proving the theorem.

Lemma 2: p **ref** $\{\alpha\} \Longleftrightarrow \forall S. \neg \exists q. (\vdash_S p \xrightarrow{\alpha} q)$

Proof

Part(i) ($\Longrightarrow$)

We construct the proof by induction on the structure of derivations that for all α, p **ref** α implies that $\forall S. \neg \exists q. (\vdash_S p \xrightarrow{\alpha} q)$. We proceed by performing a case analysis on p.

1. $\underline{p \equiv \mu.t}$

   Suppose μ.t **ref** $\{\alpha\}$, therefore $\mu \neq \alpha$. Hence $\forall S. \neg \exists q. (\vdash_S \mu.t \xrightarrow{\alpha} q)$.

2. The cases for $p \equiv nil$, $p \equiv t[\theta]$, $p \equiv (t \setminus \lambda)$ are straight forward.

3. $\underline{p \equiv t_0 + t_1}$

   Suppose $t_0 + t_1$ **ref** $\{\alpha\}$, therefore by shorter inference t_0 **ref** $\{\alpha\}$ and t_1 **ref** $\{\alpha\}$. Applying the induction hypothesis we have $\forall S. \neg \exists q. (\vdash_S t_0 \xrightarrow{\alpha} q)$ and $\forall S. \neg \exists q. (\vdash_S t_1 \xrightarrow{\alpha} q)$. Hence one cannot deduce that $\exists S, q. \vdash_S t_0 + t_1 \xrightarrow{\alpha} q$ because neither rule $(vii)^c$ nor $(viii)^c$ can ever be applied. Similarly for $p \equiv t_0 \leftthreetimes t_1$ and $p \equiv t_0 | t_1$.

4. $\underline{p \equiv fix(x.t)}$

   Let $fix(x.t)$ **ref** $\{\alpha\}$ then by shorter inference, $t[fix(x.t)/x]$ **ref** $\{\alpha\}$. Applying the induction hypothesis we can say that $\forall S. \neg \exists q. (\vdash_S t[fix(x.t)/x] \xrightarrow{\alpha} q)$. Hence since rule $(xi)^c$ always fails to apply, $\forall S. \neg \exists q. (\vdash_S fix(x.t) \xrightarrow{\alpha} q)$.

Part(ii) ($\Longleftarrow$)

We prove $\forall\alpha.(\forall S.\neg\exists q.(\vdash_S p \xrightarrow{\alpha} q) \Longrightarrow p \textbf{ ref } \{\alpha\})$ by attempting to prove $\forall\alpha.(\neg(p \textbf{ ref } \{\alpha\}) \Longrightarrow \exists S, q.(\vdash_S p \xrightarrow{\alpha} q))$. We construct the proof by induction on the structure of derivations that for all α, $\neg(p \textbf{ ref } \alpha)$ implies that $\forall S.\neg\exists q.(\vdash_S p \xrightarrow{\alpha} q))$. We proceed by performing a case analysis on p.

1. $\underline{p \equiv \mu.t}$

   Suppose $\neg(\mu.t \textbf{ ref } \{\alpha\})$, then this implies that $\mu = \alpha$. Therefore there exist an S, q such that $\vdash_S \mu.t \xrightarrow{\alpha} q$.

2. The cases for $p \equiv nil$, $p \equiv t[\theta]$, $p \equiv (t \setminus \lambda)$ are straight forward.

3. $\underline{p \equiv t_0 + t_1}$

   Suppose $\neg(t_0 + t_1 \textbf{ ref } \{\alpha\})$, then either $\neg(t_0 \textbf{ ref } \{\alpha\})$ or $\neg(t_1 \textbf{ ref } \{\alpha\})$ or both. If $\neg(t_0 \textbf{ ref } \{\alpha\})$, then by applying the induction hypothesis there exist an S, q such that $\vdash_S t_0 \xrightarrow{\alpha} q$. Therefore by rule $(vii)^c$, $\vdash_S t_0 + t_1 \xrightarrow{\alpha} q$. Similarly for the remaining sub-cases. A similar argument can be applied when $p \equiv t_0|t_1$.

4. $\underline{p \equiv t_0 \Cap t_1}$

   Suppose $\neg(t_0 \Cap t_1 \textbf{ ref } \{\alpha\})$, then either $\neg(t_0 \textbf{ ref } \{\alpha\})$ or $\neg(t_1 \textbf{ ref } \{\alpha\})$ or both. If $\neg(t_0 \textbf{ ref } \{\alpha\})$, then by applying the induction hypothesis there exist an S, q such that $\vdash_S t_0 \xrightarrow{\alpha} q$. Therefore by rule $(ix)^c$, $\vdash_S t_0 \Cap t_1 \xrightarrow{\alpha} q$. On the other hand suppose $\neg(t_1 \textbf{ ref } \{\alpha\})$, then by the induction hypothesis, there exists q for some S such that $\vdash_S t_1 \xrightarrow{\alpha} q$.We cannot guarantee that if t_0 acc A then $A \subseteq S$. Therefore unless we have rule $(xii)^c$ we cannot prove in general that for all α, $\neg(t_0 \Cap t_1)$ ref $\{\alpha\}$ implies that $\exists S, q.\ \vdash_S t_0 \Cap t_1 \xrightarrow{\alpha} q$. Nonetheless if we consider A such that t_0 acc A, and let $R = A \cup S$, then by rule $(xii)^c$ $\vdash_R t_1 \xrightarrow{\alpha} q$, and we also know that t_0 acc A and $A \subseteq R$. Hence by rule $(x)^c$ $\vdash_R t_0 \Cap t_1 \xrightarrow{\alpha} q$.

5. $\underline{p \equiv fix(x.t)}$

   Suppose $\neg$(fix(x.t) ref $\{\alpha\}$) then $\neg$(t[fix(x.t) / x] ref $\{\alpha\}$). Applying the induction hypothesis we can say that $\exists S, q.(\vdash_S t[fix(x.t)/x] \xrightarrow{\alpha} q)$. Hence by rule $(xi)^c$ we can deduce that $\vdash_S fix(x.t) \xrightarrow{\alpha} q$.

□

Lemma 3.1: $(p \textbf{ acc } A \wedge \overline{\alpha} \in A) \Longrightarrow \exists S, q.(\vdash_S p \xrightarrow{\alpha} q)$

Proof

We construct the proof by induction on the structure of derivations that for all α, if p acc A and $\overline{\alpha} \in A$ then this implies that $\exists S, q.(\vdash_S p \xrightarrow{\alpha} q)$. We proceed by performing a case analysis on p.

1. $\underline{p \equiv \mu.t}$

   Suppose $\mu.t$ acc A and $\overline{\alpha} \in A$, then $\mu = \alpha$ and $A = \{\overline{\alpha}\}$. Hence we can say that there exist S, q such that $\vdash_S \mu.t \xrightarrow{\alpha} q$.

2. The cases for $p \equiv nil$, $p \equiv t[\theta]$, $p \equiv t \setminus \lambda$ are straight forward.

3. $\underline{p \equiv t_0 + t_1}$

   Suppose $t_0 + t_1$ acc A and $\overline{\alpha} \in A$, then if t_0 acc A_0 and t_1 acc A_1 such that $A = A_0 \cup A_1$, this implies that either $\overline{\alpha} \in A_0$ or $\overline{\alpha} \in A_1$ or both. Consider the case when $\overline{\alpha} \in A_0$; applying the induction hypothesis we can say that $\exists S, q.(\vdash_S t_0 \xrightarrow{\alpha} q)$ and therefore by rule $(vii)^c$ we can deduce that $\vdash_S t_0 + t_1 \xrightarrow{\alpha} q$. Similarly for the other two cases. We can use a similar argument when $p \equiv t_0|t_1$.

4. $\underline{p \equiv t_0 \between t_1}$

   Suppose $t_0 \between t_1$ acc A and $\overline{\alpha} \in$ A, then if t_0 acc A_0 and t_1 acc A_1 such that $A = A_0 \cup A_1$, this implies that either $\overline{\alpha} \in A_0$ or $\overline{\alpha} \in A_1$ or both. Consider the case when $\overline{\alpha} \in A_0$; applying the induction hypothesis we can say that $\exists S, q.(\vdash_S t_0 \xrightarrow{\alpha} q)$ and therefore by rule $(vii)^c$ we can deduce that $\vdash_S t_0 \between t_1 \xrightarrow{\alpha} q$. Now consider the case when $\overline{\alpha} \in A_1$; applying the induction hypothesis we can say that $\exists S, q.(\vdash_S t_1 \xrightarrow{\alpha} q)$. Using a similar argument as the one used in the proof of the second part of Lemma 2, let $R = S \cup A_0$ then by rule $(xii)^c$ $\vdash_R t_1 \xrightarrow{\alpha} q$, hence by rule $(x)^c$ $\vdash_R t_0 \between t_1 \xrightarrow{\alpha} q$. The remaining case is a combination of the previous two cases.

5. $\underline{p \equiv fix(x.t)}$

   Suppose $fix(x.t)$ acc A then by shorter inference, $t[fix(x.t)/x]$ acc A. Assuming $\overline{\alpha} \in A$ we can apply the induction hypothesis, therefore $\exists S, q.(\vdash_S t[fix(x.t)/x] \xrightarrow{\alpha} q)$. Hence by rule $(xi)^c$, $\vdash_S fix(x.t) \xrightarrow{\alpha} q$.

□

Lemma 3.2: $(\exists S, q.(\vdash_S p \xrightarrow{\alpha} q) \wedge (p \text{ acc } A)) \Longrightarrow \overline{\alpha} \in A$

Proof

We construct the proof by induction on the structure of derivations that for all α if there exists S, q such that $\vdash_S p \xrightarrow{\alpha} q$ and p acc A then this implies that $\overline{\alpha} \in A$

1. $\underline{p \equiv \mu.t}$

   Suppose $\exists S, q.(\vdash_S \mu.t \xrightarrow{\alpha} q)$ and μ.t acc A then $\mu = \alpha$ and $A = \{\overline{\alpha}\}$.

2. The cases for $p \equiv$ nil, $p \equiv t[\theta]$, $p \equiv (t \setminus \lambda)$ are straight forward.

3. $\underline{p \equiv t_0 + t_1}$

Suppose $\exists S, q.(\vdash_S t_0 + t_1 \xrightarrow{\alpha} q)$ and $t_0 + t_1$ acc A such that t_0 acc A_0 and t_1 acc A_1 and $A = A_0 \cup A_1$. Since $\exists S, q.(\vdash_S t_0 + t_1 \xrightarrow{\alpha} q)$, then either $\vdash_S t_0 \xrightarrow{\alpha} q$ or $\vdash_S t_1 \xrightarrow{\alpha} q$. Consider the case when $\vdash_S t_0 \xrightarrow{\alpha} q$; applying the induction hypothesis we have $\overline{\alpha} \in A_0$, therefore $\overline{\alpha} \in A$. Similarly for the case when $\vdash_S t_1 \xrightarrow{\alpha} q$. A similar argument can be used when $p \equiv t_0 | t_1$ and $p \equiv t_0 +\!\!\!| t_1$.

4. $\underline{p \equiv fix(x.t)}$

Suppose $\exists S, q. \vdash_S fix(x.t) \xrightarrow{\alpha} q$, then by shorter inference $\vdash_S t[fix(x.t)/x] \xrightarrow{\alpha} q$. Suppose also that $fix(x.t)$ acc A, then by shorter inference $t[fix(x.t)/x]$ acc A. Applying the induction hypothesis $\overline{\alpha} \in A$.

□

Lemma 4: $(p \text{ ref } R \wedge p \sim_p q) \Longrightarrow q \text{ ref } R$.

Proof

Let us start by attempting to prove $(p \text{ ref } \{\alpha\} \wedge p \sim_p q) \Longrightarrow q \text{ ref } \{\alpha\}$. Let p ref $\{\alpha\}$, then by lemma 2 we can say that $\forall S. \neg \exists p'.(\vdash_S p \xrightarrow{\alpha} p')$. Suppose there exists q', S' such that $\vdash_{S'} q \xrightarrow{\alpha} q'$. Since $p \sim_p q$ we expect to have p' such that $\vdash_{S'} p \xrightarrow{\alpha} p'$. Clearly this is a contradiction.Therefore if p ref $\{\alpha\}$, then $\forall S. \neg \exists p'.(\vdash_S p \xrightarrow{\alpha} p')$ and if $p \sim_p q$ then $\forall S. \neg \exists q'.(\vdash_S q \xrightarrow{\alpha} q')$. Hence by lemma 2, q ref $\{\alpha\}$. If p ref R and $p \sim_p q$, then the argument above can be repeated for each member of R.

□

Lemma 5: $(p \text{ acc } A \wedge p \sim_p q) \Longrightarrow q \text{ acc } A$.

Proof

Let us attempt to prove the above lemma by proving that for all A, A', p acc A and q acc A' and $p \sim_p q$ implies that $A = A'$. Let $\overline{\alpha} \in A$, then by lemma 3.1, $\exists S, p'.(\vdash_S p \xrightarrow{\alpha} p')$. Since $p \sim_p q$ then $\vdash_S q \xrightarrow{\alpha} q'$ and $p' \sim_p q'$, therefore by lemma 3.2, $\overline{\alpha} \in A'$. The same argument can be applied for all elements of A. Therefore $A \subseteq A'$. Conversely let $\overline{\alpha} \in A'$, then by lemma 3.1 $\exists S, q'.(\vdash_S q \xrightarrow{\alpha} q')$. Since $p \sim_p q$ then $\vdash_S p \xrightarrow{\alpha} p'$ and $p' \sim_p q'$, therefore by lemma 3.2 $\overline{\alpha} \in A$. The same argument can be applied for all elements of A'. Therefore $A' \subseteq A$. Hence we can conclude that $(p \text{ acc } A \wedge p \sim_p q) \Longrightarrow q$ acc A.

□

Lemma 6: If $p \sim_p q$ then for all $\mu \in Act$, $\mu.p \sim_p \mu.q$.

Proof

Let $T = \{\langle \mu.p', \mu.q' \rangle | \; p' \sim_p q', \; \mu \in Act\} \cup \{\langle p', q' \rangle | \; p' \sim_p q'\}$. We want to show that T is a bisimulation, therefore it is sufficient to show that if $\vdash_R \mu.p \xrightarrow{\mu'} p'$ then $\exists q'.(\vdash_R \mu.q \xrightarrow{\mu'} q')$ and $\langle p', q' \rangle \in T$. The only case to consider is when $\mu = \mu'$. The proof is trivial.

□

Lemma 7: If $p \sim_p q$ then for all $s \in P$, $p|s \sim_p q|s$.
Proof
Let $T = \{\langle p'|s, q'|s\rangle |\ p' \sim_p q',\ s \in P\}$. Clearly T is symmetric; we want to show that T is a bisimulation. It is sufficient to show that if $\langle p'|s, q'|s\rangle \in T$ and $\vdash_R p'|s \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R q'|s \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$. Let us consider the following cases:

1. Let $\vdash_R p'|s \xrightarrow{\mu} p'''|s$. By shorter inference $\vdash_R p' \xrightarrow{\mu} p'''$ and s **ref** R. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$ and $p''' \sim_p q'''$. Clearly $\vdash_R q'|s \xrightarrow{\mu} q'''|s$ and by construction $\langle p'''|s, q'''|s\rangle \in T$.

2. Let $\vdash_R p'|s \xrightarrow{\mu} p'|s'$. By shorter inference $\vdash_R s \xrightarrow{\mu} s'$ and p' **ref** R. We know that $p' \sim_p q'$, therefore by lemma 4 q' **ref** R. Hence $\vdash_R q'|s \xrightarrow{\mu} q'|s'$, and by construction $\langle p'|s', q'|s'\rangle \in T$.

3. Let $\vdash_R p'|s \xrightarrow{\mu} p'''|s'$. In this case $\mu = \tau$ and therefore by shorter inference, there exists α such that $\vdash_{R_0} p' \xrightarrow{\alpha} p'''$, $\vdash_{R_1} s \xrightarrow{\overline{\alpha}} s'$, p' **ref** R_1, s **ref** R_0 for any R_0, R_1 such that $R_0 \cup R_1 = R$. Now if $\vdash_{R_0} p' \xrightarrow{\alpha} p'''$ and $p' \sim_p q'$ this implies that there exists q''' such that $\vdash_{R_0} q' \xrightarrow{\alpha} q'''$ and $p''' \sim_p q'''$. Since $p' \sim_p q'$ and p' **ref** R_1 then by lemma 4, q' **ref** R_1. Therefore we can say that $\vdash_R q'|s' \xrightarrow{\tau} q'''|s'$ and that $\langle p'''|s', q'''|s'\rangle \in T$.

□

Lemma 8: If $p \sim_p q$ then for all $s \in P$, $p + s \sim_p q + s$.
Proof
Let $T = \{\langle p' + s, q' + s\rangle |\ p' \sim_p q', s \in P\} \cup \{\langle p', q'\rangle |\ p' \sim_p q'\}$. Clearly T is symmetric; we want to show that T is a bisimulation. It is sufficient to show that if $\langle p' + s, q' + s\rangle \in T$ and $\vdash_R p' + s \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R q' + s \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$. Let us consider the following cases:

1. Let $\vdash_R p' + s \xrightarrow{\mu} p'''$, such that $\vdash_R p' \xrightarrow{\mu} p'''$. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$, and $p''' \sim_p q'''$. Clearly by rule $(vii)^c$ $\vdash_R q' + s \xrightarrow{\mu} q'''$ and by construction $\langle p''', q'''\rangle \in T$.

2. Let $\vdash_R p' + s \xrightarrow{\mu} s'$, such that $\vdash_R s \xrightarrow{\mu} s'$. By rule $(viii)^c$, $\vdash_R q' + s \xrightarrow{\mu} s'$ and by construction $\langle s', s'\rangle \in T$.

□

Lemma 9: If $p \sim_p q$ then for all $s \in P$, $p \leftthreetimes s \sim_p q \leftthreetimes s$.

Proof

Let

$$T = \{\langle p' \leftthreetimes s, q' \leftthreetimes s\rangle, \langle s \leftthreetimes p', s \leftthreetimes q'\rangle \mid p' \sim_p q', s \in P\} \cup \{\langle p', q'\rangle \mid p' \sim_p q'\}.$$

We want to show that T is a bisimulation. Therefore we have to show:

1. If $\langle p' \leftthreetimes s, q' \leftthreetimes s\rangle \in T$ and $\vdash_R p' \leftthreetimes s \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R q' \leftthreetimes s \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$.

2. If $\langle s \leftthreetimes p', s \leftthreetimes q'\rangle \in T$ and $\vdash_R s \leftthreetimes p' \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R s \leftthreetimes q' \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$.

Proof of case (i)

1. Let $\vdash_R p' \leftthreetimes s \xrightarrow{\mu} p'''$, such that $\vdash_R p' \xrightarrow{\mu} p'''$. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$, and $p''' \sim_p q'''$. Clearly by rule $(ix)^c$ $\vdash_R q' \leftthreetimes s \xrightarrow{\mu} q'''$ and by construction $\langle p''', q'''\rangle \in T$.

2. Let $\vdash_R p' \leftthreetimes s \xrightarrow{\mu} s'$, such that $\vdash_R s \xrightarrow{\mu} s'$ and p' acc $A \subseteq R$. We know that $p' \sim_p q'$, and by lemma 5 since p' acc A then q' acc $A \subseteq R$. Therefore by rule $(x)^c$, $\vdash_R q' \leftthreetimes s \xrightarrow{\mu} s'$ and by construction $\langle s', s'\rangle \in T$.

Proof of case (ii)

1. Let $\vdash_R s \leftthreetimes p' \xrightarrow{\mu} p'''$, such that $\vdash_R p' \xrightarrow{\mu} p'''$ and s acc $A \subseteq R$. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$, and $p''' \sim_p q'''$. Therefore by rule $(x)^c$ $\vdash_R s \leftthreetimes q' \xrightarrow{\mu} q'''$ and by construction $\langle p''', q'''\rangle \in T$.

2. Let $\vdash_R s \leftthreetimes p' \xrightarrow{\mu} s'$, such that $\vdash_R s \xrightarrow{\mu} s'$. Then by rule $(ix)^c$, $\vdash_R s \leftthreetimes q' \xrightarrow{\mu} s'$ and by construction $\langle s', s'\rangle \in T$.

□

Lemma 10: If $p \sim_p q$ then for all $\lambda \in \Lambda$, $p \setminus \lambda \sim_p q \setminus \lambda$.

Proof

Let $T = \{\langle p' \setminus \lambda, q' \setminus \lambda\rangle \mid p' \sim_p q', \lambda \in \Lambda\}$. We want to show that T is a bisimulation, therefore it is sufficient to show that if $\langle p' \setminus \lambda, q' \setminus \lambda\rangle \in T$ and $\vdash_R p' \setminus \lambda \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R q' \setminus \lambda \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$. Let $\vdash_R p' \setminus \lambda \xrightarrow{\mu} p''' \setminus \lambda$, then by shorter inference $\vdash_R p' \xrightarrow{\mu} p'''$ and $\mu \neq \lambda, \overline{\lambda}$. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$, and $p''' \sim_p q'''$. Therefore by rule $(ii)^c$ $\vdash_R q' \setminus \lambda \xrightarrow{\mu} q''' \setminus \lambda$ and by construction $\langle p''' \setminus \lambda, q''' \setminus \lambda\rangle \in T$.

□

Lemma 11: If $p \sim_p q$ then for all relabellings θ, $p[\theta] \sim_p q[\theta]$.
Proof
Let $T = \{\langle p'[\theta], q'[\theta]\rangle |\ p' \sim_p q',\ \theta : Act \mapsto Act\}$. We want to show that if $\langle p'[\theta], q'[\theta]\rangle \in T$ and $\vdash_R p'[\theta] \xrightarrow{\mu} p''$ then there exists q'' such that $\vdash_R q'[\theta] \xrightarrow{\mu} q''$ and $\langle p'', q''\rangle \in T$. Let $\vdash_R p'[\theta] \xrightarrow{\mu} p'''[\theta]$, such that $\vdash_R p' \xrightarrow{\mu} p'''$. We know that $p' \sim_p q'$, therefore there exists q''' such that $\vdash_R q' \xrightarrow{\mu} q'''$, and $p''' \sim_p q'''$. Therefore by rule $(iii)^c$, $\vdash_R q'[\theta] \xrightarrow{\mu} q'''[\theta]$ and by construction $\langle p'''[\theta], q'''[\theta]\rangle \in T$.
□

Definition 2
Let x be a free variable in the terms t and u, then we say $t \sim_p u$ if $t[p/x] \sim_p u[p/x]$ for all agents p.

Lemma 12: If t $\sim_p u$ then $fix(x.t) \sim_p fix(x.u)$.
Proof
Let $T = \{\langle G[fix(x.t)/x], G[fix(x.u)/x]\rangle |\ t \sim_p u, FV(G) \subseteq \{x\}\}$. We are going to construct the proof by induction on the structure of derivations that if $\vdash_R G[fix(x.t)/x] \xrightarrow{\mu} p$ then there exists q such that $\vdash_R G[fix(x.u)/x] \xrightarrow{\mu} q$ and $\langle p, q\rangle \in T$. We proceed by performing a case analysis on G.

1. $\underline{G \equiv x}$

   Let $G \equiv x$ then $\vdash_R fix(x.t) \xrightarrow{\mu} p'$, so by shorter inference $\vdash_R t[fix(x.t)/x] \xrightarrow{\mu} p'$ and by induction $\vdash_R t[fix(x.u)/x] \xrightarrow{\mu} q'$ with $\langle p', q'\rangle \in T$. By definition 3, since $t \sim_p u$, then $t[fix(x.u)/x] \sim_p u[fix(x.u)/x]$. Therefore there exists q'' such that $\vdash_R u[fix(x.u)/x] \xrightarrow{\mu} q''$ and $\langle p', q''\rangle \in T$. Now $G[fix(x.u)/x] \equiv fix(x.u)$ when $G \equiv x$ and by the recursion rule $(xi)^c$ $\vdash_R fix(x.u) \xrightarrow{\mu} q''$. Therefore we are done.

2. $\underline{G \equiv fix(y.v), y \not\equiv x}$

   By assumption we have $\vdash_R fix(y.(v[fix(x.t)/x])) \xrightarrow{\mu} p'$, then by shorter inference $\vdash_R (v[fix(x.t)/x][G[fix(x.t)/x]/y]) \xrightarrow{\mu} p'$, which may be rewritten as $\vdash_R v[G/y][fix(x.t)/x] \xrightarrow{\mu} p'$. So by induction applied to the expression $v[G/y]$, we know that $\vdash_R (v[G/y][fix(x.u)/x]) \xrightarrow{\mu} q'$, with $\langle p', q'\rangle \in T$. By manipuating substitutions and applying the recursion rule $(xi)^c$ we obtain $\vdash_R G[fix(x.u)/x] \xrightarrow{\mu} q'$ as required.

□

Theorem 1: $\sim_p$ is a congruence with respect to the operators of our language.
Proof
Lemmas 6 – 12 together make up this property.
□

6 Results

Figure 3 below shows some equivalence preserving syntactic transformations based on the notion of strong bisimulation as presented in this paper.

$P + Q \sim_p Q + P$
$P + P \sim_p P$
$P|Q \sim_p Q|P$
$P|(Q|R) \sim_p (P|Q)|R$
$P[f][f'] \sim_p P[f' \circ f]$
$nil \nmid P \sim_p P$
$\alpha.t_0 \nmid \alpha.t_1 \sim_p \alpha.t_0$

$P + (Q + R) \sim_p (P + Q) + R$
$P + nil \sim_p P$
$P|nil \sim_p P$
$P[Id] \sim_p P$
$P \nmid nil \sim_p P$
$\tau.t_0 \nmid t_1 \sim_p \tau.t_0$

All the laws shown above may be proved by exhibiting appropriate strong bisimulations. The proofs are not included in this paper, but, they follow the same arguments found in [8].

7 Conclusions

In this paper we have investigated an alternative approach to introducing the notion of priority to process algebras. We do not claim that the approach presented here is the best approach, but, we hope that it will lead to further discussion on the subject. Further work needs to be done – we would like to axiomatize a weak observational equivalence based on our semantics and investigate whether our priority operator can be included in a synchronous calculus such as SCCS [12].

8 Acknowledgements

I express my thanks to Prof. Glynn Winskel and Dr. Alan Mycroft for encouraging me to work on the subject and for their advice and suggestions. I am also grateful to Dr. Mike Gordon and Andy Gordon for their comments and feedback on the work. Thanks are also due to Trinity College who are kindly supporting my stay at Cambridge.

References

[1] Baeten, J.C.M, Bergstra J.A. and Klop J.W,. Syntax and Defining Equations for an interrupt mechanism in Process Algebra. Report CS-R8503, Center for Mathematics and Comp. Sci, Amsterdam, Feb. 1985.

[2] Bergstra J.A. and Klop J.W. Process Algebra for Synchronous Communication. Information and Control 60, 1984, pp. 109-137.

[3] G.Boudol, I.Castellani. Concurrency and Atomicity. INRIA SOPHIA-ANTIPOLIS 06560 Valbonne, France.

[4] Juanito Camilleri. An operational semantics for occam, (EXTENDED VERSION). Computing Lab, University of Cambridge. Technical Report $n°144$. August 1988.

[5] R.Cleaveland, M.Hennessy. Priorities in Process Algebras. Department of Computer Science, University of Sussex. Report $n°2/88$. March 1988.

[6] Costa, G., Stirling, C., Weak and strong fairness in CCS. pp. 245-264, *Mathematical Foundations of Computer Science*, ed. M.P. Chytil, V. Koubek, LNCS 176, Springer 1984.

[7] Edsger.W.Dijkstra. A Discipline of Programming. Prentice-Hall International Series in Automatic Computation.

[8] Hennessy. M and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. Journal of the ACM 32, $n°1$, January 1985, pp137-161.

[9] C.A.R.Hoare. Communicating Sequential Processes. Prentice-Hall International Series in Computer Science.

[10] INMOS ltd. occam Programming Manual. Prentice-Hall International Series in Computer Science.

[11] Robin Milner. A Calculus of Communicating Systems. Lecture notes in Computer Science. Springer-Verlag series $n°92$.

[12] Robin Milner. Calculi for Synchrony and Asynchrony. Department of Computer Science, Edinburgh University. February 1982.

[13] Gordon.D.Plotkin. A Structural Approach to Operational Semantics. Department of Computer Science, Aarhus University Denmark. Sept 1981.