# A Model for the $\pi$- Calculus *

M. Hennessy
University of Sussex

## Abstract

We develop a semantic theory based on testing for a minor variant of the $\pi$-calculus. The resulting semantic equivalence can be characterised using of acceptance sets and can also be characterised as an equational theory. We define a class of interpretations for the $\pi$-calculus and construct one which is

# 1 Introduction

In [MPW92a], [MPW92b], a calculus of mobile processes, the $\pi$-calculus, is presented. The first reference is an introduction to the calculus and the second develops a semantic theory based on bisimulations, [Mil89]. The $\pi$-calculus is an extension of the process algebra

where it is written as $t \setminus x$. In fact it is with this operator that the communication of private channels may be represented in the language. Unlike the original $\pi$-calculus we use an *if* ... *then* ... *else* ... statement. The process *if be then t else u* acts like $t$ if the boolean expression *be* is true and like $u$ otherwise. We allow a very simple language of boolean expressions. Essentially it allows the testing of identity between channel names. Finally we have recursive definitions; intuitively the process $rec X.\ t$ is equivalent to a process $X$ where $X$ has been defined by the equation $X = t$. We use $\Sigma$ to denote the set of operators $\{nil, \Omega, (x), \overline{x}y, +, \oplus, |\}$, i.e. all the operators except prefixing by input actions; the interpretation of this last operator will require special attention.

As usual *rec* acts like a binder for process variables and we have an appropriate

[t]

Input $\qquad \dfrac{-}{x(y).p \xrightarrow{x(v)} p\{v/y\}} \quad v \notin fn((y)p)$

Output $\qquad \dfrac{-}{\overline{x}y.p \xrightarrow{\overline{x}y} p}$

Sum $\qquad \dfrac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$

Par $\qquad \dfrac{p \xrightarrow{a} p'}{p \mid q \xrightarrow{a} p' \mid q} \quad bn(a) \cap fn(q) = \emptyset$

Res $\qquad \dfrac{p \xrightarrow{a} p'}{(y).p \xrightarrow{a} (y).p'} \quad y \notin n(a)$

Open $\qquad \dfrac{p \xrightarrow{\overline{x}y} p'}{(y).p \xrightarrow{\overline{x}(v)} p'\{v/y\}} \quad y \neq x$ and $v \notin fn((y)p')$

If $\qquad \dfrac{p \xrightarrow{a} p', \quad \llbracket be \rrbracket = tt}{\mathit{if}\, be \ \mathit{then}\ p \ \mathit{else}\ q \ \xrightarrow{a} p'}$

$$\dfrac{q \xrightarrow{a} q', \quad \llbracket be \rrbracket = \mathit{ff}}{\mathit{if}\, be \ \mathit{then}\ p \ \mathit{else}\ q \ \xrightarrow{a} q'}$$

Figure 1: Rules for external actions

In the remainder of this section we develop some technical results about this transition system and it may be skipped by the reader uninterested in technical details. Most of the proofs are omitted as they are rather tedious and usually proceed by syntactic analysis. The first two results are taken directly from [MPW92b] and the proofs can be transferred directly to our language.

**Lemma 2.1** *If $p \rightarrowtail p'$ then $fn(p') \subseteq fn(p)$ and if $p \xrightarrow{a} p'$ then $fn(p') \subseteq fn(p) \cup bn(a)$ and $fn(a) \subseteq fn(p)$* □

**Definition 2.2** We use the phrase

if $p \xrightarrow{a} p'$ then *equally* $q \xrightarrow{a} q'$

to mean that if $p \xrightarrow{a} p'$ may be inferred from the transition rules then $q \xrightarrow{a} q'$ may also be inferred, with an inference of no greater depth. We use a similar notation for internal arrows. □

**Lemma 2.3** *Suppose that $p \xrightarrow{\alpha(y)} p'$ where $\alpha = x$ or $\alpha = \overline{x}$ and that $z \notin n(p)$. Then equally $p \xrightarrow{\alpha(z)} p''$ for some $p''$ such that $p'' \equiv_\alpha p'\{z/y\}$.* □

[t]

Ω
$$\frac{-}{\Omega \succ\!\!\!\longrightarrow \Omega}$$

Rec
$$\frac{-}{recX.\ t \succ\!\!\!\longrightarrow t[recX.\ t/X]}$$

Choice
$$\frac{p \succ\!\!\!\longrightarrow p'}{p \oplus q \succ\!\!\!\longrightarrow p'}$$

Op
$$\frac{p_i \succ\!\!\!\longrightarrow p_i'}{op(\ldots, p_i, \ldots) \succ\!\!\!\longrightarrow op(\ldots, p_i', \ldots)} \quad \text{for } \ op \in \{+, |, (y)\}$$

Com
$$\frac{p \xrightarrow{\overline{x}y} p', \quad q \xrightarrow{x(z)} q'}{p \mid q \succ\!\!\!\longrightarrow p' \mid q'\{y/z\}}$$

Close
$$\frac{p \xrightarrow{\overline{x}(w)} p', \quad q \xrightarrow{x(w)} q'}{p \mid q \succ\!\!\!\longrightarrow (w)(p' \mid q')}$$

If
$$\frac{p \succ\!\!\!\longrightarrow p', \quad [\![be]\!] = tt}{if\ be\ then\ p\ else\ q \quad \succ\!\!\!\longrightarrow p'}$$
$$\frac{q \succ\!\!\!\longrightarrow q', \quad [\![be]\!] = f\!f}{if\ be\ then\ p\ else\ q \quad \succ\!\!\!\longrightarrow q'}$$

Figure 2: Rules for internal transitions

In the operational semantics bound names are liberally renamed and it is important to establish that $\alpha$-conversion does not seriously affect the behavioural properties of processes. The next series of results have this in mind. In the next section a new kind of action, a *free input* action, will be seen to be important and therefore these results will be also established for these actions. Recall that $p \xrightarrow{x(y)} p'$ means that $y$ is acting as a reference in $p'$ to where names which are received by $p$ are placed. This is actually how it is used in the rule Com of Figure 2. So it is natural to define a free input action by

$p \xrightarrow{xy} p'\{y/z\}$ whenever$is$

then $u$ must be of the form $\overline{x}(z).u'$ for some $z$ such that $s \equiv_\alpha u'\{y/z\}$. We will make frequent use of the notion of a *harmless* action or sequence of actions. In a particular statement a sequence of actions is considered harmless if the bound variables are different than any of the variables appearing in the rest of the statement. This definition is of the course

**Proof:** Once more both results are proved simultaneously by induction on the length of the derivations. Note that in i) it is sufficient to sim

**Lemma 2.8** *For any $a, p$ the set $D(p, a)$ is finite.*

**Proof:** The proof is straightforward by structural induction on $p$. Note that if $p$ has the form $rec\,X.\,t$ then this set is empty for every $a$. $\square$

Of course a process may

# 3 Testing Processes

Given the operational semantics of the previous section we may now apply the standard theory of testing as developed in [Hen88]. To this end we assume a special name called $\omega$ which is used to denote success. A *test* or *experiment* is then simply a process which may use this extra name and applying a test $e$ to a process $p$ consists in running the process $e \mid p$ to completion. A *computation* from $e \mid p$ is a complete sequence of the form

$$e \mid p = s_0 \rightarrowtail s_1 \rightarrowtail \ldots \rightarrowtail s_k \rightarrowtail \ldots$$

i.e. it is either infinite or if $s_n$ is the last element there must be no $s'$ such that $s_n \rightarrowtail s'$. Such a computation is *successful* if some $s_k$ can report success, i. e. $s_k \xrightarrow{\overline{\omega}(x)}$ for any name $x$. We often write this as $s_k \in \mathit{Succ}$. Then we write

$$p \ \mathit{must} \ e$$

if every computation from $p \mid e$ is successful. Finally we say

$$p \underset{\sim}{\sqsubset} q$$

if for every experiment $e$, $p \ \mathit{must} \ e$ implies $q$

**Definition 3.1** For every sequence $s \in RAct$

1. let $\downarrow s$ be defined by

   - $p \downarrow \varepsilon$ if there is no infinite internal computation from $p$, i.e. no infinite computation of the form $p \rightarrowtail p_1 \rightarrowtail \ldots \rightarrowtail p_k \rightarrowtail \ldots$.
   - $p \downarrow a.s$ if $p \downarrow$ and for every $p'$ such that $p \overset{a}{\Longrightarrow} p'$ $p' \downarrow s$.

2. let $p \Downarrow s$ if for every $s'$ such that $s \equiv_\alpha s'$ $p \downarrow s'$.

□

To prove that the latter is preserved by $\alpha$-conversion we need a lemma.

**Lemma 3.2** *For every process $p$*

1. *$p\sigma \downarrow$ implies $p \downarrow$*
2. *if $\sigma$ is injective on $fn(p)$ then $p \downarrow$ implies $p\sigma \downarrow$*
3. *$p \equiv_\alpha q$ and $p \downarrow$ implies $q \downarrow$.*

**Proof:** As an example we prove the second statement. Suppose

$$p\sigma \rightarrowtail r_1 \rightarrowtail \ldots \rightarrowtail r_n \rightarrowtail \ldots$$

is an infinite sequence. By Lemma 2.5 $p \rightarrowtail p_1$ such that $p_1\sigma \equiv_\alpha r_1$ and by Proposition 2.7, part ( ), $p_1\sigma \rightarrowtail r_2'$ such that $r_2 \equiv_\alpha r_2'$. Again by Lemma 2.5 $p_1 \rightarrowtail p_2$ such that $p_2\sigma \equiv_\alpha r_2$. Continuing in this way we obtain an infinite internal computation from $p$. □

As a corollary we have

**Proposition 3.3**

1. *If $p \equiv_\alpha q$ and $p \Downarrow s$ then $q \Downarrow s$.*
2. *if $\sigma$ is injective on $fn(p)$ then $p \Downarrow s$ implies $p\sigma \Downarrow \sigma(s)$*
3. *if $p\sigma \Downarrow s'$ then for every $\sigma'$ and $s$ such that $p\sigma \equiv_\alpha p\sigma'$ and $\sigma'(s) \equiv_\alpha s'$, $p \Downarrow s$.*

**Proof:** The first statement directly from the previous lemma and the results at the end of the previous section. For suppose $p \Uparrow s$, i. e. for some prefix $s'$ of some $s''$ such that $s \equiv_\alpha s''$ $p \overset{s'}{\Longrightarrow} p'$ such that $p \uparrow$. So by Proposition 2.7 $q \overset{s_1'}{\Longrightarrow} q'$ for some $q'$ and $s_1'$ such that $s' \equiv_\alpha s_1'$ and $q' \equiv_\alpha p'\{\underline{bn}(s')/\underline{bn}(s)\}$. Since $p' \uparrow$ it follows from the previous lemma that $p'\{\underline{bn}(s')/\underline{bn}(s)\} \uparrow$ and therefore that $q' \uparrow$. This in turn implies that $q \Uparrow s$.

We leave the second statement to the reader and concentrate on the last. Suppose $p \Uparrow s$, i. e. $p \overset{u}{\Longrightarrow} p'$ where $p \uparrow$ for some prefix $u$ of $s$. By Proposition 2.7 this means that $p\sigma' \overset{u'}{\Longrightarrow} r$ for some $r$ and $u'$ such that $r \equiv_\alpha p'\sigma'[\underline{bn}(u) \mapsto \underline{bn}(u')]$ and $u' \equiv_\alpha \sigma'(u)$. From the previous lemma this means that $r \uparrow$ and therefore $p\sigma' \Uparrow u'$, i. e. $p\sigma' \Uparrow s'$. Applying clause (1) we obtain $p\sigma \Uparrow s$. □

In fact in order to establish that a process is convergent with respect to all sequences which are $\alpha$-equivalent to $s$ it is sufficient to establish it for a sequence whose bound variables are new, i.e. a harmless sequence.

**Lemma 3.4** *If $bn(s) \cap fn(p) = \emptyset$ then $p \Downarrow s$ if and only if $p \downarrow s$.*

**Proof:**

- for some harmless $s'$ such that $s \equiv_\alpha s'$, (i. e. $bn(s') \cap (fn(p) \cup fn(q)) = \emptyset$), $\mathcal{A}(p, s') \ll \mathcal{A}(q, s')$

□

First we show that this preorder is preserved by $\alpha$-conversion.

**Proposition 3.8** *If* $p \equiv_\alpha q$ *then* $p \ll q$.

**Proof:** From Proposition . we know that that $p \equiv_\alpha q$ and $p \Uparrow s$ implies $q \Uparrow s$ and the result therefore follows by the preceding lemma. □

**Proposition 3.10** *If $p \ll q$ then $p \sqsubseteq_{\sim} q$.*

**Proof:** The proof has the same structure as that for the corresponding result in [Hen88], Lemma 4.4.1 , although the details are more complicated because of the different forms of communication allowed in the $\pi$-calculus. Suppose $p \ll q$ and $p$ *must* $e$. We show $q$ *must* $e$ by examining an arbitrary computation from $e \mid q$ :

$$e \mid q = r_0 \rightarrowtail r_1 \rightarrowtail \ldots \rightarrowtail r_k \rightarrowtail \ldots \qquad (*)$$

and proving that there is some $e_n$ such that $e_n \in Succ$. The proof depends on whether the computation $(*)$ is finite or infinite. As an example we consider only the finite case. So we may assume that $r_k$ is stable for some $k$. Each $r_i$ is of the form $(\underline{v}_i)r'_i$ where the individual restricted names in the sequence $\underline{v}_i$ arise because of the possible use of the Close rule from the operational semantics. Nevertheless by concentrating on the interaction between the two processes the computation $(*)$ may be unzipped into two derivations from $e$, $q$ respectively, which use only actions from $Act$ and which show their individual contributions:

$$e = e_0 \overset{\overline{a}_1}{\Longrightarrow} \ldots e_j \ldots \overset{\overline{a}_k}{\Longrightarrow} e_m$$

and

$$q = q_0 \overset{a_1}{\Longrightarrow} \ldots q_j \ldots \overset{a_k}{\Longrightarrow} q_m.$$

These are such that for each $j$ there exists an $i$ such that $r'_i = e_j \mid q_j$ and if $a_i = x(v), \overline{a}_i = \overline{x}(v)$ then $r_{i-1} \rightarrowtail r_i$ is inferred using an instance of the rule Close. If on the other hand it is inferred using an instance of the rule Com then if $a_i = {}^{-}$

one involving a harmless subsequence and combine this with a corresponding derivation from $e$ to obtain an unsuccessful computation from $e \mid p$.
When the computation $(*)$ is infinite the only possibility we have not touched on is when the unzipped derivations are infinite and $p$ converges on all subsequences. Here we make use of Corollary 2.10 which states that the computation trees from $p$ and $q$ are finite branching, modulo $\alpha$-conversion. The details of how this is used may be found in Lemma 4.4.1 from [Hen88]. $\square$

To prove the converse we need to define two sets of special tests one of which tests for convergence and the other which is capable of testing for the contents of $\mathcal{A}(p, s)$. The crucial point is to be able to distinguish between outputing a free name on a channel and outputing an internal link. To achieve this we use the fact that if a process outputs a free name this name must belong to the free names of the process. We first examine convergence.

Let $X$ be a finite set of names. For each $s$ in $Aprow$

1. The rule Com is used in the derivation.
   Then $r$ has the form $p' \mid \mathit{if}\, y \in X \;\mathit{then}\, \omega \;\mathit{else} \ldots$ where $p \xrightarrow{\overline{x}y} p'$. This means that $y \in fn(p) \subseteq X$ and therefore $r\sqrt{succ}$.

2. The rule Close is used in the derivation.
   Here $r$ has the form

$$(y)(p' \mid (\; \mathit{if}\, y \in X \;\mathit{then}\, \omega \;\mathit{else}\; c(s'\{z/y\})_{X \cup \{z\}} \;)\{y/z\})$$

   for some new name $z$, where $p \xrightarrow{\overline{x}(y)} p'$. If $y \in X$ then this term is obviously in $\sqrt{succ}$. If not we know that $p' \Downarrow s'$ because $p \Downarrow s$ and since $fn(p') \subseteq X \cup \{y\}$ we may apply induction to obtain that $p' \mid c(s')_{X \cup \{y\}}\sqrt{succ}$. But by the previous lemma $c(s')_{X \cup \{y\}} \equiv_\alpha c(s'\{z/y\})_{X \cup \{z\}})\{y/z\}$ and therefore $c(s')_{X \cup \{y\}} \mid p' \equiv_\alpha c(s'\{z/y\})_{X \cup \{z\}})\{y/z\} \mid p'$. It now follows that $r\sqrt{succ}$.

Conversely suppose $p \mid c(s)_X\sqrt{succ}$. Obviously $p\downarrow$ and to show $p \Downarrow s$ it is sufficient to prove that if $p \overset{\overline{x}(y)}{\Longrightarrow} p'$ then $p \Downarrow s'$. Now it may not be possible for $c(s)_X$ to perform $\overline{x}(y)$ because $y$ may be in $X$. So pick a completely new $v$. Then $p \overset{\overline{x}(v)}{\Longrightarrow} p'\{v/y\}$ and $p \mid c(s)_X \rightarrowtail r$ where up to $\alpha$-conversion we may take $r$ to be

$$(v)p'\{v/y\} \mid \; \mathit{if}\, v \in X \;\mathit{then}\, \omega \;\mathit{else}\; c(s'\{v/y\})_{X \cup \{v\}} \;.$$

Moreover we know that $r\sqrt{succ}$ and therefore that $p'\{v/y\} \mid c(s'\{v/y\})_{X \cup \{v\}}\sqrt{succ}$. By induction this means $p'\{v/y\} \Downarrow s'\{v/y\})$. The simple substitution $\{y/v\}$ is injective on $fn(p')$ and so we may apply Proposition . to conclude $p'\{v/y\}\{y/v\} \Downarrow s'\{v/y\}\{y/v\}$, i.e. $p' \Downarrow s'$. □

We next design a test $e(s, B)_X$, where $s \in RAct^*$ and $B$ a finite subset of $\mathcal{N}$ with the property that whenever $p \Downarrow s$ and $fn(p) \subseteq X$

$$p \;\mathit{must}\; e(s, B)_X \iff \forall A \in \mathcal{A}(p, s) \;\; B \cap A \neq \emptyset.$$

Note that the right hand side is trivially satisfied if $\mathcal{A}(p, s) = \emptyset$. First let $e(x)$, $e(\overline{x})$ denote the tests $\overline{x}y.\overline{\omega}y.nil, x(y).\overline{\omega}y.nil$ respectively, for any name $y$. Then we define $e(s, B)_X$ by induction on $s$:

1. $e(\varepsilon, B)_X = \sum\{\, e(y) \mid y \in B\,\}$

2. $e(xy.s, B)_X = 1.w + \overline{x}y.e(s, B)_{X \cup \{y\}}$

. $e(\overline{x}y.s, B)_X = 1.w + x(z).\; \mathit{if}\, z = y \;\mathit{then}\; c(s, B)_X \;\mathit{else}\; \omega$ where $z$ is a new name

4. $e(\overline{x}(y).s, B)_X = 1.w + x(z).\; \mathit{if}\, z \in X \;\mathit{then}\, \omega \;\mathit{else}\; e(s\{z/y\}, B)_{X \cup \{z\}}$ where $z$ is a new name.

**Proposition 3.13** *If $p \Downarrow s$ and $fn(p) \subseteq X$ then*

$$p \;\mathit{must}\; e(s, B)_X \iff \forall A \in \mathcal{A}(p, s) \;\; B \cap A \neq \emptyset.$$

**Proof:** The proof is by induction on $s$ and again we examine only one case, when $s$ has the form $\overline{x}(y).s'$

First suppose that $p \mid e(s,B)_X\sqrt{succ}$ and $A \in \mathcal{A}(p,s)$. We must show that $B \cap A \neq \emptyset$. We know that $p \overset{x(y)}{\Longrightarrow} p'' \overset{s'}{\Longrightarrow} p'$ for some stable $p'$ such that $A = Subj(p')$. Because $y$ may appear free in the test $e(s.B)_X$ we may not be able to use $y$ in a communication between the process and the test. So choose a new $v$ and by Proposition 2.7 we have, up to $\alpha$-conversion, $p \overset{x(v)}{\Longrightarrow} p''\{v/y\} \overset{s\{v/y\}}{\Longrightarrow} p'\{v/y\}$. Moreover by Lemmas 2.4 and 2.5 and Proposition 2.6 it follows that $A\{v/y\} = Subj(p'\{v/y\})$. Because $v$ is new we now have that, again up to $\alpha$-conversion,

$$p \mid e(s,B)_X \rightarrowtail^* (v)p''\{v/y\} \mid \textit{ if } v \in X \textit{ then } \omega \textit{ else } e(s'\{v/y\}, B\{v/y\})_{X \cup \{v\}} \ .$$

Here we have used an analogue to Lemma .11 for the tests, namely that $(e(s,B)_X)\sigma \equiv_\alpha e(\sigma(s), B\sigma)_{X\sigma}$. From this it follows that $p''\{v/y\} \mid e(s'\{v/y\}, B\{v/y\})_{X\cup\{v\}})\sqrt{succ}$. So by induction $A' \cap B\{v/y\} \neq \emptyset$ for every $A' \in \mathcal{A}(p''\{v/y\}, s'\{v/y\})$. One such $A'$ is $A\{v/y\}$ and so $A \cap B \neq \emptyset$, because $v$ is new.

Conversely suppose that for all $A \in \mathcal{A}(p,s)$ $A \cap B \neq \emptyset$. We show that $p \mid e(s,B)_X\sqrt{succ}$. We know that $p \downarrow$ and the proof proceeds by induction on this fact. Suppose $p \mid e(s,B)_X \rightarrowtail r$. We must show that $r\sqrt{succ}$. If this move is because of an internal move of either the process or the test we can apply induction or else the result follows trivially by the construction of the tests. So we need only consider the case when there is communication between the process and the test. We consider the case when this is because of an application of the rule Close. The other possibility, when the rule Com is used, is left to the reader. Then $r$ must have the form

$$(v)(p' \mid \textit{ if } v \in X \textit{ then } \omega \textit{ else } e(s'\{v/y\}, B\{v/y\})_{X\cup\{v\}} \ ),$$

up to $\alpha$-conversion, where $p \xrightarrow{x(v)} p'$. It is sufficient to consider the case when $v$ is not in $X$ when effectively any continuing computation is from $p' \mid e(s'\{v/y\}, B\{v/y\})_{X\cup\{v\}}$. So the result will follow by induction if we can show that for every $A' \in \mathcal{A}(p', s'\{v/y\})$ $A' \cap B\{v/y\} \neq \emptyset$. One can show that any such $A'$ has the form $A\{v/y\}$ where $A \in \mathcal{A}(p,s)$. Since $A \cap B \neq \emptyset$ this implies $A' \cap B\{v/y\} \neq \emptyset$. □

With these two proposition we can now prove the converse of Proposition .10 and therefore the alternative characterisation of $\sqsubseteq\!\!\!\!\sim$ .

**Theorem 3.14** *For every pair of processes $p, q$, $p \sqsubseteq\!\!\!\!\sim\ q$ if and only if $p \ll q$.*

**Proof:** We need only prove $p \ll q$ implies $p \sqsubseteq\!\!\!\!\sim\ q$ and this follows directly from the previous two propositions. For example suppose that $p \Downarrow s$, $q \Downarrow s$ and $B \in \mathcal{A}(q, s')$ where $s'$ is new. We derive a contradiction from the assumption that for all $A \in \mathcal{A}(p,s')$ $A \not\subseteq B$. For each such $A$ there must be some $x_A$ in $A$ and not in $B$. Let $L = \{\, a_A \mid A \in \mathcal{A}(p,s) \,\}$ and choose $X$ so that it contains both $fn(p)$ and $fn(q)$. Then $p$ *must* $e(s', L)_X$ whereas $q$ need not always pass $e(s', L)_X$ and this contradicts the fact that $p \sqsubseteq\!\!\!\!\sim\ q$. □

This theorem also shows that the behavioural preorder $\sqsubseteq\!\!\!\!\sim$ is determined by a small collection of tests, namely all those of the form $e(s, B)_X$ or $c(s)_X$. We call this set of tests *CTest* and they will be used in the next section.

As an application of the alternative characterisation we show that $\sqsubseteq\!\!\!\!\sim$ is preserved by most of the operators of the language.

**Proposition 3.15** *For every operator op in* $\Sigma$ $p_i \sqsubseteq\!\!\!\!\sim q_i$ *implies* $op(\ldots, p_i, \ldots) \sqsubseteq\!\!\!\!\sim op(\ldots, q_i, \ldots)$.

**Proof:** For the operator $|$ it is best to prove this directly from the definition of $\sqsubseteq\!\!\!\!\sim$ using the fact that $p \mid q$ *must* $e$ if and only if $p$ *must* $q \mid e$. For the other operators is is easier to prove the result for $\ll$. The only non-trivial case is for the binding operator $(y)-$. As an example of the proof technique let us show that if $p \ll q$ and $(y)q \Uparrow s$ then $(y)p \Uparrow s$. So without loss of generality we can suppose that $(y)q \stackrel{s}{\Longrightarrow} r$ where $r \uparrow$. If the rule Open is not used in this derivation then $r$ has the form $(y)q'$ where $q \stackrel{s}{\Longrightarrow} q'$. So $q \Uparrow s$ from which it follows that $p \Uparrow s$ and therefore $(y)p \Uparrow s$ since $y$ can not appear in $s$. So suppose Open is used. Then the derivation can be viewed as

$$(y)q \stackrel{s_1}{\Longrightarrow} (y)q_1 \xrightarrow{\overline{x}(v)} q_1'\{v/y\} \stackrel{s_2}{\Longrightarrow} r$$

where

$$q \stackrel{s_1}{\Longrightarrow} q_1 \xrightarrow{\overline{x}y} q_1'$$

and $v \not\forall\}$

## 4 Modelling the Language $L_\pi$

In this section we address the

Given such a natural interpretation, $D$, we can define a semantic interpretation of $L_\pi$ following the usual approach of denotational semantics. We let $Env_D$ be the set of $D$-environments, i.e. mappings from $PV$ to $D$, ranged over by $\rho$ and we assume an evaluation function $[\![\ ]\!]: BExp \longmapsto \{tt, f\!f\}$. Then the semantics of the language $L_\pi$ is given as a function:

$$D[\![\ ]\!]: L_\pi \longmapsto (Env_D \longmapsto D)$$

and is defined by structural induction:

$$\text{i)}\quad D[\![X]\!]\rho = \rho(X)$$

$$\text{ii)}\quad D[\![op(\underline{t})]\!]\rho = op_D(D[\![\underline{t}]\!]\rho)$$

$$\text{iii)}\quad D[\![recP.t]\!]\rho = Y\lambda d.D[\![t]\!]\rho[d/P]$$

$$\text{iv)}\quad D[\![\ \mathit{if}\, be\ \mathit{then}\, t\ \mathit{else}\ u\ ]\!]\rho \;=\; \begin{array}{l} D[\![t]\!]\rho\ \text{ if }\ [\![be]\!] = tt \\ D[\![u]\!]\rho\ \text{ if }\ [\![be]\!] = f\!f \end{array}$$

$$\text{v)}\quad D[\![x(y).t]\!]\rho = in_D(x, \lambda y.D[\![t]\!]\rho)$$

where $Y$ is the least-fixpoint operator for

$$\begin{aligned}
X \oplus (Y \oplus Z) &= (X \oplus Y) \oplus Z \\
X \oplus Y &= Y \oplus X \\
X \oplus X &= X \\
X + (Y + Z) &= (X + Y) + Z \\
X + Y &= Y + X \\
X + X &= X \\
X + nil &= X \\
pre.X + pre.Y &= pre.(X \oplus Y) \\
x(y).X + x(y).Y &= x(y).X \oplus x(y).Y \\
\overline{x}y.X + \overline{x}y'.Y &= \overline{x}y.X \oplus \overline{x}y
\end{aligned}$$

the principal being a version of the interleaving law. Unlike the standard theories of concurrency, such as that in [Mil89], the restriction operator can not be eliminated from all finite terms using the equations; this is a reflection of the extra power of restriction in the $\pi$-calculus. However the irreducible occurrences can be coded up as a form of derived prefix.

**Definition 4.6** If $x \neq y$ then $\overline{x}(y)p$ is a shorthand for the term $(y)\overline{x}y.p$ and the subject of the prefix $\overline{x}(y)$ is $\overline{\ }$

$$(X \oplus Y) \mid Z = X$$

**Lemma 4.9** *If $p \downarrow$ then there exists a hnf, hnf $(p)$, such that $\vdash_r p = hnf\,(p)$ and $\mid p \mid = \mid hnf(p) \mid$.*

**Proof:** It is virtually identical to that of Proposition 4.2.1 of [HI91] and is therefore omitted. The only new ingredient is that in the subterms $p_{\overline{x}}$ there is at most one summand of the form $\overline{x}(y).p'$. If during the reduction procedure two such summands are generated then they can be replaced by one using the equations as follows:

$$\begin{aligned} \overline{x}(y).p' + \overline{x}(z).p'' &= \overline{x}.(w)p'\{w/y\} + \overline{x}.(w)p''\{w/z\} \quad \text{by } \alpha\text{-conversion, where } w \text{ is new} \\ &= (w)( \end{aligned}$$

I $$\frac{}{p \le p} \qquad\qquad \frac{p \le q,\ q \le r}{p \le r}$$

II $$\frac{p_i \le q_i, \quad 1 \le i \le n}{op(p_1, \ldots, p_n) \le op(q_1, \ldots, q_n)} \quad \text{for every } op \in \{\overline{x}y., (x), +, \oplus, |\}$$

Eq $$\frac{}{p \le q} \quad \text{for every instance of an inequation}$$

$\alpha$ $$\frac{p \equiv_\alpha q}{p \le q}$$

Input $$\frac{p\{z/y\} \le q\{z/y\} \ \text{ for all names } \ z}{x(y).p \le x(y).q}$$

If1 $$\frac{p \le p'}{\mathit{if}\, be \ \mathit{then}\ p \ \mathit{else}\ q \quad \le \quad \mathit{if}\, be \ \mathit{then}\ p' \ \mathit{else}\ q}$$

$$q \le q'$$

operations are defined pointwise:

$$op(\ldots, I, \ldots) = \{\, op_{P_E}(\ldots, e, \ldots) \mid e \in I \,\} \downarrow$$

where $S \downarrow$

and it follows that $C_E[\![x(y).b]\!] = \{[x(y).b]\}\downarrow$ . □

**Proposition 4.15** *For every $b \in BF$ and process $q$ $\vdash_r b \leq q$ implies $C_E[\![b]\!] \leq C_E[\![q]\!]$.*

**Proof:** The proof follows from the property

> $\vdash_r b \leq q$ implies that there exists a $d \preceq q$ such that
> for some $m \geq 0$ $\vdash b \leq d^{(m)}$. $(*)$

This is proved by induction on the length of the proof of $\vdash_r b \leq q$ and proceeds by considering the last rule applied in the proof. There are only three non-trivial cases, the Input rule, the Unwind rule and the Transitivity rule. As an example we look at the Input rule. Here $b, q$ have the form $x(y).b', x(y).p$ respectively and $\vdash_r b \leq q$ has been inferred because for all $z \Rightarrow$

$\vdash_r b$

**Lemma 4.18** *For every test $e \in CTest$ $p$ must $e$ implies that there exists some $b \in BF$*

for any $f{:}\mathcal{N} \longmapsto C_E$.
Let $f = \bigvee f^n$ where $f^n$ is defined such that for all $x \notin \mathcal{N}^n$ $f^n(x) = \Omega$ and for all $x \in \mathcal{N}^n$ $f^n(x)$ is a compact element. This is possible because $C_E$ is an algebraic cpo. Then $in_{C_E}(x, f) = \bigvee_n in_{C_E}(x, f^n)$. It is not difficult to show that for each $n$

$$in_{C_E}(x, f^n) = \{[x(w).\ \mathit{if}\, w \in \mathcal{N}^n\ \mathit{then}\ f^n(w)\ \mathit{else}\, \Omega\ \ ]\} \downarrow$$

and therefore

$$in_{C_E}(x, f) = \bigvee_n \{[x(w).\ \mathit{if}\, w \in \mathcal{N}^n\ \mathit{then}\ f^n(w)\ \mathit{else}\, \Omega\ \ ]\} \downarrow .$$

So

$$\begin{aligned} i_D(in_{C_E}(x, f)) &= \bigvee_n D[\![x(w).\ \mathit{if}\, w \in \mathcal{N}^n\ \mathit{then}\ f^n(w)\ \mathit{else}\, \Omega\ \ ]\!] \\ &= \bigvee_n in_D(x, \lambda w.\ \mathit{if}\, w \in \mathcal{N}^n\ \mathit{then}\ D[\![f^n(w)]\!]\ \mathit{else}\, \bot\ \ ) \\ &= \bigvee_n in_D(x, i_D \cdot f^n) \\ &= in_D(x, i_D \cdot f). \end{aligned}$$

□

These results show that at least there are reasonable models of the language and as a byproduct we have a sound and complete proof system for the behavioural preorder. This is obtained by adding $\omega$-induction to the proof system. Note that one can also replace the infinitary Input rule with the finitary one suggested by Proposition .16 and retain completeness. However $CI_E$, the initial fully-abstract model constructed in this section, is a term model and it would be more satisfactory if we had an independent description of it, for example as some modification of the acceptance trees in [Hen88]. The main difficulty here is to find a version of these trees which will support a reasonable definition of the restriction operator $(y)$.

Another deficiency in this section is the general definition of what constitutes an interpretation of the language. It would be more satisfactory if this took into consideration the fact that the operator $(y)$ also binds names. So in addition to having a special way of interpreting the input operator, using the functions $in_D$, we would also have a special function for restriction. One suggestion would be to have a function $res_D$ of type $(\mathcal{N} \longmapsto D) \longmapsto D$ and then to define $D[\![(y)t]\!]\rho$ to be $res_D(\lambda y.[\![t]\!]\rho)$. With this definition $\alpha$-conversion would be sound in all interpretations. However it is difficult to extend the results of this section to this new form of interpretation. It seems that a more subtle interpretation of restriction is required and one possibility is to adopt the approach taken in [Win88].

# References

[BD92] M. Boreale and R. DeNicola. Testing for mobile processes. In *Proceedings of CONCUR 92*, 1992.