A Fully Abstract Denotational Semantics for the π-Calculus*

Matthew Hennessy
School of Cognitive and Computing Sciences
Unive sity of Sussex

June 26, 1996

Abstract

This paper describes the construction of two set-theoretic denotational models for the π-calculus. The models are obtained as initial solutions to domain equations in a functor category. By associating with each syntactic construct of the π-calculus a natural transformation over these models we obtain two interpretations for the language.

We also show that these models are *fully abstract* with respect to natural behavioural preorders over terms in the language. By this we mean that two terms are related behaviourally if and only if their interpretations in the model are related. The behavioural preorders are the standard versions of *may* and *must* testing adapted to the π-calculus. S

1 Introduction

The π-calculus [15, 16] is a process algebra for describing processes which communicate by exchanging *channel name* . These names may be *private* to a particular process but such a process may decide to share a name with certain other processes by exporting it. This ability of processes to share common private channel names is the main source of the expressive power of the π-calculus, at least in relation to other *value-pa ing* process algebras.

Numerous semantic theories have been proposed for the π-calculus, and it variants, [16, 21, 24, 4]. However, at least until very recently, all of these theories were *behaviour* based, i.e. an operational semantics is first given to the language and then a semantic theory is developed by abstracting, using a variety of methods, from certain aspects of this operational view of processes. Thus, for example, in [21] *late* and *early* bisimulation equivalences are developed for the π-calculus and are characterised using a proof system for establishing identities between processes. A similar programme is carried out in [4] for an appropriate adaptation of *te ting equivalence* [7].

*This work was supported by the EU EXPRESS Working Group and the Royal Society. Much of the research reported here was carried out during a visit to INRIA, Sophia-Antipolis.

whose objects are finite subsets of $\mathcal{N}$ and whose morphisms are injections $s \stackrel{\sigma}{\longmapsto} s'$. Let $\mathcal{D}$ be some suitable category of *domain* . Then $\mathcal{D}^I$ is the category whose

1. objects are functors from I to $\mathcal{D}$
2. morphisms are natural transformations

The domain $\mathbf{P}$ we use is an object in $\mathcal{D}^I$, for a suitable choice of $\mathcal{D}$.

A particular domain $\mathbf{P}_s$, from this uniform family of domains, needs to be sufficiently rich to describe the behaviour of all processes with free names included in s. However it should be clear that this description will require the use of names which are not in s. For example even if the free names of the process p, of the form $n\,?\,\lambda\, x.\, t$, are included in s a complete description of its behaviour will have to include what happens when it receives along the channel n a name which is not in s. Denotationally this will be reflected in the fact that the domain equation characterising $\mathbf{P}$ will have an input component of the form $\mathbf{N} \times (\mathbf{N} \Rightarrow \mathbf{P})$. Here $\mathbf{N}$, the representation of the set of names $\mathcal{N}$ is the trivial functor whose action at s returns s itself, and $\Rightarrow$ is the exponential operator in the functor category. This is such that the action of $(\mathbf{N} \Rightarrow \mathbf{P})$ at s describes not only a function from s to $\mathbf{P}_s$ but also a *uniform* collection of functions from s' to $\mathbf{P}_{s'}$ for each s' such that $s \stackrel{\sigma}{\longmapsto} s'$. This is more than sufficient to capture the consequences of inputting names not in s.

In a similar manner the proper treatment of restriction requires that the description of $\mathbf{P}_s$ includes names not in s. whoseinins

6T13309.aiour will5.13.91990Td2860.20TdeT13309.cessT135

form of tensor product which we denote by . In short $l\mathcal{SL}^I$ is an *autonomou* category, i.e. a closed symmetric monoidal category.

We use as our domain the initial solution to the equation

$$\begin{aligned}
\mathbf{P} &\cong (\mathbf{A} \times (\mathbf{N} \ {}^{\top} \mathbf{F}) \times (\mathbf{N} \ {}^{\top} \mathbf{C}))_{\perp} \\
\mathbf{F} &\cong \mathbf{N} \Rightarrow \mathbf{P} \\
\mathbf{C} &\cong (\mathbf{N} \quad \mathbf{P}) + \mathbf{s}(\mathbf{P})
\end{aligned}$$

in the category $l\mathcal{SL}^I$. Intuitively a non-trivial object in $\mathbf{P}$ has three components,

- $\mathbf{A}$, a functor representing the initial nondeterministic behaviour of a process
- $\mathbf{N}$

In the next section, Section 4 we first present the general requirements necessary for a domain $\mathbf{P}$ to provide an interpretation for the language. This is followed by the description of the interpretation in any such domain. The main point of this section is to give the definition of a particular model, satisfying these requirements, as the initial solution to the domain equation given above in the $l\mathcal{SL}^I$.

Section 5 contains the main result of the paper. The model $\mathbf{P}$ provides for all s-environments ρ, i.e. mappings from $\mathcal{N}$ to s, an interpretation $[\![p]\!]_s\rho$ of the process term p in the domain $\mathbf{P}_s$. We prove

for all process terms p, q, $p\rho \sqsubseteq_{must} q\rho$ if and only if $[\![p]\!]_s\rho \leq [\![q]\!]_s\rho$,

i.e the model is *fully ab tract* with respect to the behavioural preorder $\sqsubseteq_{must}$. This section first gives an overview of the proof of this result, reducing it to three independent theorems. These are tackled in three subsequent subsections. The main technical device is that of an *acceptance*. This consists of a finite sequence of *action* , appropriate to the π-calculus, followed either by Ω, indicating an ability to diverge, or by a finite set of *communication potential* , a communication potential taking the form either $n?$ or $n!$.

- The first result is an *internal full ab traction* result for the model. Sets of acceptances can be associated with objects in the domains $\mathbf{P}_s$ and we prove that equality in the model is determined by these sets.

- Sets of acceptances can also be associated behaviourally with process terms of the language, using the operational semantics. The second result, in Subsection 5.2, states that this behavioural set of acceptances coincides with that associated with the interpretation of the term in the model.

- The final result shows that the behavioural preorder $\sqsubseteq_{must}$ is also determined by sets of acceptances, those associated behaviourally with process terms.

The paper ends with a brief section, Section 6, outlining similar results for *may te ting*. A minor variation on the domain equation is solved in the category $I^{u\mathcal{SL}}$, where $u\mathcal{SL}$ is the category of upper semi-lattices, and an outline of a proof that the resulting interpretation is fully abstract with respect to $\sqsubseteq_{may}$.

We end this introduction with a brief comparison with other research. As we have already stated, most of the semantic investigations into the π-calculus has been based on behavioural theories of processes. Those presented in [15, 16] are based on suitable adaptations of *bi imulation equivalence*, [14], and these are characterised equationally, for subsets of the π-calculus, in [21]. Similarly in [4, 11], the testing preorders from [7] are adapted to the π-calculus and equationally characterised. In [4] the standard

The models constructed in the present paper are set theoretic, in the sense that they are defined using more or less standard domain theoretic constructions, although the categories in which the domain equations are solved are rather complicated. The techniques used are borrowed directly from [25] where a fully abstract model is constructed for the recursion-free sublanguage of the π-calculus. The domain used is the initial solution in the category SFP^I of a domain equation which is a simple variation on the domain equation used in [1] to model *CCS* with respect to late strong bisimulation. No explicit definition is given for the natural transformations which interpret language constructs but the use of the category SFP^I, and the natural transformation $\mathbf{s}$ is well-motivated from a monadic view of computation, as expounded in [18].

A very similar approach is taken in [19] where a very similar model to that in [25] is constructed for the same subset of the π-calculus with respect to the same equivalence, strong late bisimulation equivalence, although the same model is used to characterise the associated congruence. However here the emphasis is more on exhibiting a general framework which can be instantiated to

$$
\begin{aligned}
(x+y)+z &= x+(y+z) \\
x+0 &= x \\
x+(y \wedge z) &= (x+y) \wedge (x+z) \\
x \wedge (y+z) &= (x \wedge y)+(x \wedge z) \\
x \wedge y &\leq x+y
\end{aligned}
$$

i.e. the extra structure is that of a

singleton set. Let $l\mathcal{D}_p$ denote the category whose objects are functors from I to $l\mathcal{SL}_p$ and whose morphisms are natural transformations. There are of course the usual variations on this, for example $l\mathcal{D}, u\mathcal{D}_p, u\mathcal{D}$, obtained by using $l\mathcal{SL}$, $u\mathcal{SL}_p$, $u\mathcal{SL}$ respectively, in place of $l\mathcal{SL}_p$. These are autonomous categories as they inherit much of the structure of the underlying categories. Products, coproducts and are defined pointwise but as usual exponentiation, $\Rightarrow$, the right-adjoint of , must be expressed in terms of natural transformations. The objects in $(D \Rightarrow E)_s$, i.e. $(D \Rightarrow E)$ acting on the set s, are dependent families of objects

$$\prod_{s \overset{i}{\longmapsto} s'} (D_{s'} \Rightarrow E_{s'})$$

satisfying certain uniformity constraints, while the morphisms between exponentials are determined by the requirement for $\Rightarrow$ to be functorial. Luckily we will only have a restricted need for exponentiations.

Let $\mathcal{A}$ be the object in $l\mathcal{D}_p$ whose action on the finite set s gives the free (lower) choice predomain $\mathcal{A}(s+s)$, where $s+s$ is the disjoint union of s with itself, and which acts on the morphism $s \overset{i}{\longmapsto} s'$ to give the obvious continuous function from $\mathcal{A}_s$ to $\mathcal{A}_{s'}$, defined by $\mathcal{A}_{(s \overset{i}{\longmapsto} s')} A = \{\, i(a) \mid a \in A \,\}$. The object $\mathbf{N}$ is defined analogously: on the set s it gives the free object of $l\mathcal{SL}_p$ generated by s, i.e. the set of subsets of s, and on morphisms it simply generalises injections to sets. To construct our model we will only require exponentials of the form $(\mathbf{N} \Rightarrow A)$ which have a particularly simple form; $(\mathbf{N} \Rightarrow A)_s$ can be represented by a collection of functions over $\mathcal{N}$.

Let $\mathcal{F}_s(A)$ denote the set of all functions f with domain $\mathcal{N}$ such that

1. $f(n) \in A_{s \cup \{n\}}$
2. for all n, m not in s, $f(n) = A_{(id + m \mapsto n)} f(m)$.

Under the standard pointwise ordering this is a predomain which is a representation of the predomain $(\mathbf{N} \Rightarrow A)_s$.

We wish to solve a domain equation in $l\mathcal{D}$ which involves finding the initial solution to a functor over it. Note that $l\mathcal{SL}$ is a *cpo-enriched* category, i.e. in the terminology of [9] it is an $\mathcal{O}$-category, and this is inherited pointwise by $l\mathcal{D}$. This means that any *locally continuou* functor, [9], will have an initial fixpoint. In our domain equation we will use the locally continuous functors $\times$, and $\Rightarrow$ together with the very simple one $\mathbf{s}$. It acts on a functor A as follows:

$$\begin{aligned} (\mathbf{s}\, A)_s &= A_{(s+1)} \\ (\mathbf{s}\, A)_{(s \overset{i}{\longmapsto} s')} &= A_{(i+1)} \end{aligned}$$

where $i+1$ is the obvious morphism in I from $s+1$ to $s'+1$. For any functor A we use up_A to denote the embedding of A into $\mathbf{s}(A)$, i.e. up_A is the natural transformation between A and $\mathbf{s}(A)$ which at s is given by the morphism A_{in} from A_s to $A_{(s+1)}$ where $s \overset{in}{\longmapsto} s+1$ denotes the injection of s into $s+1$.

The functor $\mathbf{s}$ satisfies the following useful properties:

$$\begin{aligned} \mathbf{s}(A^{\top}) &\simeq (\mathbf{s}(A))^{\top} \\ \mathbf{s}(A_{\perp}) &\simeq (\mathbf{s}(A))_{\perp} \end{aligned}$$

$$
\begin{array}{rcl}
\mathbf{s}(A \times B) & \simeq & \mathbf{s}(A) \times \mathbf{s}(B) \\
\mathbf{s}(A + B) & \simeq & \mathbf{s}(A) + \mathbf{s}(B) \\
\mathbf{s}(\mathbf{N} \quad A) & \simeq & (\mathbf{N} \quad \mathbf{s}(A)) + \mathbf{s}(A) \\
\mathbf{s}(\mathbf{N} \Rightarrow A) & \simeq & \mathbf{s}(\mathbf{N}) \Rightarrow \mathbf{s}(A)
\end{array}
$$

Notation: In general we use $a\colon A \to B$ to denote the fact that, in the category under consideration, a is a morphism from the object A to the object B. In the particular case of the underlying category I we use $A \stackrel{a}{\longmapsto} B$ and sometimes when working in a functor category we will use $a\colon A \stackrel{\cdot}{\longrightarrow} B$ to emphasise that the morphisms are natural transformations. We will always use $(A \Rightarrow B)$ to indicate the exponential in a category, i.e. the object consisting of all morphisms from A to B.

3 The Language

Let $\mathcal{V}$ be a set of process variables, ranged over by $X, Y, \ldots$, and $\mathcal{N}$ a countable set of names ranged over by $x, y, n, m, \ldots$. Then the syntax of terms is given by:

$$
\begin{array}{rcl}
t \in Exp & ::= & 0 \mid n?f \mid n!x.t \mid \nu(n).t \\
 & & \mid t \parallel t \mid t + t \mid t \oplus t \\
 & & \mid \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ t \mid X \mid \mathsf{rec}\,\mathsf{X}.t \\
f \in Abs & ::= & \lambda x.t \\
b \in Bool & ::= & x = y \mid \neg b \mid b \wedge b \mid b \vee b
\end{array}
$$

There are two binding operators for names and one for process variables. Terms may have free occurrences of both and these are determined by the simple inference system given in Figure 1. The judgements are of the form $\Gamma \vdash t : A$ where Γ is a list of names and process variables which may occur free in t. The type of terms may be P, for processes or F for abstractions. As is standard we use notation such as $\Gamma + n$ to indicate the list obtained from Γ by adding n, assuming that n does not already appear in Γ. A name substitution is a function over $\mathcal{N}$ and we assume the standard definition of the application of a substitution σ to a term t to give the new term $t\sigma$; this renames bound names as required in order to avoid the capture of substituted names.

The operational semantics of the language is given as a reduction relation $\stackrel{\tau}{\longrightarrow}$ between closed terms, terms p such that $\underline{x}$

$$\frac{}{\underline{x}; X_1 \ldots X_i \ldots X_n \vdash X_i : P} \qquad \frac{\ldots \alpha\ \beta \ldots \vdash t : P}{\ldots \beta\ \alpha \ldots \vdash t : P}$$

$$\frac{}{\Gamma \vdash 0 : P} \qquad \frac{\Gamma + n \vdash t : P}{\Gamma \vdash \nu(n)\,.\,t : P}$$

$$\frac{\Gamma + n \vdash f : F}{\Gamma + n \vdash n\,?\,f : P} \qquad \frac{\Gamma + n + x \vdash t : P}{\Gamma + n + x \vdash n\,!\,x\,.\,t : P}$$

$$\frac{\Gamma \vdash t : P \quad \Gamma \vdash u : P}{\Gamma \vdash t\ op\ u : P}\ op = \|, +, \oplus$$

$$\frac{\Gamma \vdash t : P \quad \Gamma \vdash u : P}{\Gamma \vdash \mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ u : P}$$

$$\frac{\Gamma + x \vdash t : P}{\Gamma \vdash \lambda x.t : F} \qquad \frac{\Gamma + X \vdash t : P}{\Gamma \vdash \mathsf{rec}\,\mathsf{X}\,.\,t : P}$$

Figure 1: Free variables in terms

some evaluation mechanism for boolean expressions, which in view of their simplicity, should be obvious.

As an example of an application of the reduction rules consider the process term

$$p = (\nu(y)\,.\,n\,!\,y\,.\,r + n\,!\,x\,.\,r') \parallel n\,?\,(\lambda z.\,z\,!\,y\,.\,0 + z\,!\,x\,.\,0)$$

Up to α-equivalence there are two possible reductions from p, the first to $r' \parallel (x\,!\,y\,.\,0 + x\,!\,x\,.\,0)$ by the transmission of the name x, and the second to $\nu(w)\,.\,r[w/y] \parallel (w\,!\,y\,.\,0 + w\,!\,x\,.\,0)$ where w is some new name.

We refer the reader to papers such as [23, 17] for evidence of the expressive power of this language. Here we confine our attention to defining two behavioural preorders over process terms and in later sections we build fully abstract denotational models for these preorders.

The behavioural preorders are based directly on the ideas of *te ting* as developed for example in [7, 10]. A process p is tested (to conform to some behavioural requirement) by running it in parallel with another process e, presumably designed with the behavioural requirement in mind. This test is deemed a success if the experimenter, e, reaches a specified state. To model this *ucce tate* we introduce a *new* name ω, not occuring in $\mathcal{N}$ and we say e is in the success state if it can emit an output along ω, we indicate this by the notation $e \xrightarrow{\omega}$

Definition 1 *Let p mu t e if for every maximal computation*

$$e \parallel p = c_0 \xrightarrow{\tau} c_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} c_k \xrightarrow{\tau} \ldots$$

Axioms:

$$\overline{n\,?\,\lambda\,x\,.\,p \xrightarrow{n?m} p[m/x]} \qquad \overline{n\,!\,m\,.\,p \xrightarrow{n!m} p}$$

$$\overline{p \oplus q \xrightarrow{\tau} p} \qquad \overline{p \oplus q \xrightarrow{\tau} q}$$

$$\overline{\mathsf{rec}\,\mathsf{X}\,.\,t \xrightarrow{\tau} t[\mathsf{rec}\,\mathsf{X}\,.\,t/X]}$$

Communication Rule:

$$\frac{p \xrightarrow{n?m} p',\ q \xrightarrow{n!m} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'} \qquad \frac{p \xrightarrow{n!m} p',\ q \xrightarrow{n?m} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

Context Rules:

$$\frac{p \xrightarrow{\mu} p'}{p \parallel q \xrightarrow{\mu} p' \parallel q} \qquad \frac{q \xrightarrow{\mu} q'}{p \parallel q \xrightarrow{\mu} p \parallel q'}$$

$$\frac{p \xrightarrow{\tau} p'}{p + q \xrightarrow{\tau} p' + q} \qquad \frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'}\ \alpha = n?m,\ n!m$$

$$\frac{p \xrightarrow{\mu} p'}{\nu(x)\,.\,p \xrightarrow{\mu} \nu(x)\,.\,p'} \qquad x \text{ not in } \mu$$

$$\frac{[\![b]\!] = true,\ p \xrightarrow{\mu} p'}{\mathsf{if}\ b\ \mathsf{then}\ p\ \mathsf{else}\ q \xrightarrow{\mu} p'} \qquad \frac{[\![b]\!] = fal\ e,\ q \xrightarrow{\mu} q'}{\mathsf{if}\ b\ \mathsf{then}\ p\ \mathsf{else}\ q \xrightarrow{\mu} q'}$$

$$\frac{p \equiv p',\ p' \xrightarrow{\mu} q',\ q' \equiv q}{p \xrightarrow{\tau} q}$$

Structural Equivalence:

$$\begin{aligned} (\nu(x)\,.\,p) \parallel q &\equiv \nu(x)\,.\,(p \parallel q) \text{ provided } x \notin fv(q) \\ p &\equiv q \text{ provided } p \equiv_\alpha q \\ (p \parallel q) \parallel r &\equiv p \parallel (q \parallel r) \\ p + q &\equiv q + p \end{aligned}$$

Figure 2: Operational Semantics

there i ome n uch that $c_n \xrightarrow{\omega}$

Then $p \sqsubseteq_{must} q$ *if for every te t e,* p *mu t e implie q mu t e.*

This preorder is based on the idea of a process guaranteeing certain behaviour. There is a weaker preorder which captures the idea of processes being capable of certain be-

haviour: $p \sqsubseteq_{may} q$ if for every test e, p *may* e implies q *may* e, where q *may* e means that there is some maximal computation from $e \parallel q$ which reaches a state c_n such that $c_n \xrightarrow{\omega}$.

In the next Section we build a model which provides a denotational semantics which is fully abstract with respect to $\sqsubseteq_{must}$ and in Section 6 we show how this can be modified so as to provide a semantics which is fully abstract with respect to $\sqsubseteq_{may}$.

We end this section by noting that both of these preorders are natural generalisations of Morris style contextual preorders, [3, 13]. We have restricted the possible testing contexts to be of the simple form $e \parallel [\ ]$. We conjecture that the model can also be used to provide a denotational semantics which is fully abstract with respect to a notion of testing where more general contexts are allowed, although it is essential that the action for reporting the success ω be new, i.e. an action which cannot be performed by the process being tested. A result along these lines, for *may* testing, is given at the end of Section 6.

4 The Interpretation

We interpret the language in the category $l\mathcal{D}_p$. To do so we use an object in the subcategory $l\mathcal{D}$, which we call $\mathbf{P}$, together with an appropriate morphism, i.e. a natural transformation, for each combinator in the language. In the standard way, [6], we can then associate with each typing judgement $\underline{x} \vdash p : P$ a morphism from $\mathbf{N}^{\otimes \underline{x}}$ to $\mathbf{P}$ in the category $l\mathcal{D}_p$. Because of the nature of the objects $\mathbf{N}^{\otimes \underline{x}}$ these morphisms have a simple representation in terms of *environment* . For any finite subset s of $\mathcal{N}$ let ENV_s be the set of mappings from $\mathcal{N}$ to s. For $\rho, \rho' \in ENV_s$ let $\rho =_{\underline{x}} \rho'$ if $\rho(y) = \rho'(y)$ for every name y in the vector $\underline{x}$. Then the set of morphisms from $(\mathbf{N}^{\otimes \underline{x}})_s$ to $\mathbf{P}_s$, in the category $l\mathcal{SL}_p$, is in one-one correspondence with the set of mappings from ENV_s to $\mathbf{P}_s$ which preserve $=_{\underline{x}}$. In other words our semantics is equivalent to a family of interpretations

$$[\![\ \ ]\!]_s : ENV_s \to \mathbf{P}_s$$

Moreover naturality ensures that this is a *uniform* family of interpretations; the family of domains $\{\, \mathbf{P}_s \mid s \subseteq \mathcal{N} \,\}$ comes equipped with translation morphisms $(\mathbf{P}_\sigma) : \mathbf{P}_s \to \mathbf{P}_{s'}$, for each injection $s \xrightarrow{\sigma} s'$ and these translation morphisms also relate the interpretation of terms in the different domains:

$$[\![p]\!]_{s'}(\sigma \circ \rho) = (\mathbf{P}_\sigma)([\![p]\!]_s$$

- $\|_{\mathbf{P}}$, a morphism from $\mathbf{P}$ $\mathcal{P}$

To complete our description of the interpretation we must exhibit the specific choice domain $\mathbf{P}$, an object in $l\mathcal{D}$, together with the required morphisms. $\mathbf{P}$ is essentially a functorial version of the *Acceptance Tree* model of [12], adapted to the π-calculus.

We let $\mathbf{P}$ be the initial solution to the equation

$$\begin{aligned}\mathbf{P} &\cong (\mathbf{A} \times (\mathbf{N} \ {}^{\top}\!\!\to \mathbf{F}) \times (\mathbf{N} \ {}^{\top}\!\!\to \mathbf{C}))_{\perp} \\ \mathbf{F} &\cong \mathbf{N} \Rightarrow \mathbf{P} \\ \mathbf{C} &\cong (\mathbf{N} \ \to \mathbf{P}) + \mathbf{s}(\mathbf{P})\end{aligned}$$

In order to explain the intuition behind this equation let us consider the action of this functor on an arbitrary object of I, a finite subset s of $\mathcal{N}$. $\mathbf{P}_s$ is either the bottom element $\perp$ or has three components:

- $\mathbf{A}_s$, an acceptance set over $s+s$, representing the initial nondeterministic behaviour of a process. We require two copies of s in order to record both the input and output potentials of processes. To emphasise this use of $s+s$ we let $n?$, $n!$ denote $in_l(n)$, $in_r(n)$ respectively, when applied to $s+s$.

- $(\mathbf{N} \ {}^{\top}\!\!\to \mathbf{F})_s$, a finite partial function from s to $\mathbf{F}_s$, representing the potential input behaviour of a process on some finite set of input channels. The objects in $\mathbf{F}_s$ represent the functional behaviour of the process on receipt of an input. $\mathbf{F}$ is isomorphic to $(\mathbf{N} \Rightarrow \mathbf{P})$ and so, as outlined in Section 2, $\mathbf{F}_s$ is isomorphic to $\mathcal{F}_s(\mathbf{P})$. This encodes the subsequent behaviour of the process on receipt of any name in s and on receipt of an *arbitrary* name not in s.

- $(\mathbf{N} \ {}^{\top}\!\!\to \mathbf{C})_s$ another partial function from s to $\mathbf{C}_s$, representing the potential output behaviour along a finite number of output channels. The output behaviour associated with each channel, in $\mathbf{C}_s$, is of two kinds:

  - $(\mathbf{N} \ \to \mathbf{P})_s$, a finite non-empty function from s to $\mathbf{P}_s$, representing a finite set of pairs, each pair consisting of a name to be output along the channel and an element of $\mathbf{P}_s$ encoding

To define the natural transformation $\mathbf{in_P}$

as its trivial extension. So to complete the definition we must specify $\mathbf{new}_f$. Since $\mathbf{F}$ is isomorphic to $\mathbf{N} \Rightarrow \mathbf{P}$, by definition, and we have the isomorphism $\mathbf{s}(\mathbf{N} \Rightarrow \mathbf{P}) \simeq (\mathbf{s}(\mathbf{N}) \Rightarrow \mathbf{s}(\mathbf{P}))$ we can define $\mathbf{new}_f$ to be $(up_{\mathbf{N}} \Rightarrow \mathbf{new})$.

- $\mathbf{new}_3$ is defined in a similar manner as the the trivial extension of $id_{\mathbf{N}} \quad ^{\top} \mathbf{new}_c + \lambda x. \top$ where $\mathbf{new}_c$ is a natural transformation from $\mathbf{s}(\mathbf{C})$ to $\mathbf{C}$. $\mathbf{s}(\mathbf{C})$ is

Informally these can be viewed as terms in the language (with the added constant Ω); viewed in this manner $\nu(\mathtt{x})\,.$ acts as a binding operator on *acceptance* and we will only consider them up to α-conversion.

Acceptance are ordered as follows:

- $\Omega \ll \mathtt{a}$ for every *acceptance* $\mathtt{a}$
- $\mathtt{a} \ll \mathtt{b}$ implies
  - $\mathtt{n?x.a} \ll \mathtt{n?x.b}$
  - $\mathtt{n!x.a} \ll \mathtt{n!x.b}$
  - $\nu(\mathtt{x})\,.\,\mathtt{a} \ll \nu(\mathtt{y})\,.\,\mathtt{b}[\mathtt{y}/\mathtt{x}]$ for every $y \notin fv(\nu(x)\,.\,a)$.

This ordering is lifted to sets of *acceptance* by:

$\mathtt{A} \ll \mathtt{B}$ if for every $\mathtt{b} \in \mathtt{B}$ there exists some $\mathtt{a} \in \mathtt{A}$ such that $\mathtt{a} \ll \mathtt{b}$.

Sets of *acceptance* can be associated with elements of the domains $\mathbf{P}_s$ and, behaviourally , with process terms of the language. We first consider the latter and to do so we need some notation. The operational semantics is given in terms of a reduction relation $\stackrel{\tau}{\longrightarrow}$ between process terms. Here we need to describe the *potential action* a term can perform. There are three kinds of potential actions,

1. an input action $\stackrel{n?x}{\longrightarrow}$, as described in Section 3
2. an output action $\stackrel{n!x}{\longrightarrow}$, also as described in Section 3
3. a *bound* output action $\stackrel{\nu(z)\,.\,n!z}{\longrightarrow}$

We use Act, ranged over by α, to denote this set of potential actions. The bound output action may be defined by:

$$p \stackrel{\nu(z)\,.\,n!z}{\longrightarrow} q, \text{ if } p \parallel n\,?\,\lambda y\,.\,y\,!\,y\,.\,0 \stackrel{\tau}{\longrightarrow} \nu(z)\,.\,(q \parallel z\,!\,z\,.\,0), \text{ where } z \notin fv(p).$$

These are extended to sequences over Act by

- $p \stackrel{\varepsilon}{\Longrightarrow} q$ if $p \stackrel{\tau}{\longrightarrow}{}^{*} q$
- $p \stackrel{\alpha\,.\,s}{\Longrightarrow} q$ if $p \stackrel{\tau}{\longrightarrow}{}^{*} \stackrel{\alpha}{\longrightarrow} \stackrel{s}{\Longrightarrow} q$

We write $p \Uparrow$ if p *diverge* , i.e. there is an infinite sequence of derivations

$$p \stackrel{\tau}{\longrightarrow} p_1 \stackrel{\tau}{\longrightarrow} \ldots\ldots \stackrel{\tau}{\longrightarrow} p_k \stackrel{\tau}{\longrightarrow} \ldots$$

and $p \Downarrow$ if it *converge* , i.e. there is no such sequence. Finally let $S(p)$ denote the subset of the set of *indication* , Ind, defined by

$$\{\, n? \mid p \stackrel{n?x}{\Longrightarrow} \text{ for some } x \,\} \cup \{\, n! \mid p \stackrel{n!x}{\Longrightarrow} \text{ or } p \stackrel{\nu(x)\,.\,n!x}{\Longrightarrow} \text{ for some } x \,\}.$$

With this notation we can now define the behavioural acceptances associated with a process term.

- $p \models^{b} \Omega$ if $p \Uparrow$

- $p \models^b \mathtt{A}$ if $p \Downarrow$ and $A = S(q)$ for some q such that $p \stackrel{\varepsilon}{\Longrightarrow} q$
- $p \models^b \alpha.\mathtt{a}$ if $p \Downarrow$ and $q \models^b \alpha$ for some q such that $p \stackrel{\alpha}{\Longrightarrow} q$

Let $\mathtt{Acc}^b(p)$ denote the set $\{\mathtt{a} \mid p \models^b \mathtt{a}\}$. These sets characterise the behavioural preorder $\sqsubseteq_{must}$:

Theorem 5.2 *$p \sqsubseteq_{must} q$ if and only if $\mathtt{Acc}^b(p) \ll \mathtt{Acc}^b(q)$.*

The proof of this theorem is given in Sub-section 5.3.

In a similar manner we can associate with each element d of the domain $\mathbf{P}_s$ a set of acceptances $\mathtt{Acc}_s(d)$; the details are postponed until Sub-section 5.1 because the structure of $\mathbf{P}_s$ needs to be elaborated.

The following theorem is often called an *internal full ab traction* result as it gives an alternative characterisation of identity in the model.

Theorem 5.3 *For every $d, e \in \mathbf{P}_s$, $d \leq e$ if and only if $\mathtt{Acc}_s(d) \ll \mathtt{Acc}_s(e)$.*

The proof of this theorem will be given in the Sub-section 5.1.

The third major result relates the behavioural *acceptance* of a process term with the *acceptance* of its denotation. In general these will not be identical. For example, from the definition of $\mathtt{Acc}^b_s$ to be given in the next sub-section, it turns out that the only acceptance sets in $\mathtt{Acc}^b_s(n\,!\,x\,.\,0 + m\,!\,y\,.\,0)$ are $\{n!\}$ and $\{m!\}$ but $\{n!, m!\}$ is also in $\mathtt{Acc}_s([\![n\,!\,x\,.\,0 + m\,!\,y\,.\,0]\!]_s)id$, where $s = \{n, m\}$. However the sets of behavioural *acceptance* and denotational *acceptance* associated with process terms will coincide up to the kernel of $\ll$. For two such sets $\mathtt{A}$, $\mathtt{B}$ let $\mathtt{A} \cong \mathtt{B}$ if $\mathtt{A} \ll \mathtt{B}$ and $\mathtt{B} \ll \mathtt{A}$, i.e. for every $\mathtt{a} \in \mathtt{A}$ there is some $\mathtt{b} \in \mathtt{B}$ such that $\mathtt{b} \ll \mathtt{a}$ and conversely for every $\mathtt{b} \in \mathtt{B}$ there is some $\mathtt{a} \in \mathtt{A}$ such that $\mathtt{a} \ll \mathtt{b}$.

Theorem 5.4 *Suppo e $x_1 \ldots x_k \vdash p : P$. Then for every s uch that $x_i \in s$, $\mathtt{Acc}^b(p) \cong \mathtt{Acc}_s([\![p]\!]_s id)$ where $id \in ENV_s$ i any environment which i the identity on s.*

Subsection 5.2 is devoted to the proof of this Theorem.

With these three results we can now give the proof of Theorem 5.1:
Let $\underline{x}$ be such that $\underline{x} \vdash p : P$, $\underline{x} \vdash q : P$ and let s be such that $x_i \in s$ for every i. Then

$$\begin{array}{rcl} p\rho \sqsubseteq_{must} q\rho & \iff & \mathtt{Acc}^b(p\rho) \ll \mathtt{Acc}^b(q\rho) \text{ by Theorem 5.2} \\ & \iff & \mathtt{Acc}_s([\![p\rho]\!]_s id) \ll \mathtt{Acc}_s([\![q\rho]\!]_s id) \text{ by Theorem 5.4} \\ & \iff & [\![p\rho]\!]_s id \leq [\![q\rho]\!]_s id \text{ by Theorem 5.3} \\ & \iff & [\![p]\!]_s \rho \leq [\![q]\!]_s \rho \end{array}$$

The last line is an application of the *Sub titution Lemma*:

$$[\![p\rho]\!]_s id = [\![p]\!]_s (id \circ \rho).$$

5.1 Internal Full Abstraction

In this section we give the details of the internal full abstraction result for the model $\mathbf{P}$. In order to associate *acceptance* with elements of the domains $\mathbf{P}_s$ we first need to develop some notation for describing their various components.

As outlined in Section 2 for any functor $\mathbf{H}$ the predomain $(\mathbf{N} \stackrel{\top}{\Rightarrow} \mathbf{H})_s$ can be represented as $(s \multimap \mathbf{H}_s)$. Moreover if $\sigma: \mathbf{H} \stackrel{\cdot}{\longrightarrow} \mathbf{K}$ then the action of the natural transformation $(id \stackrel{\top}{\Rightarrow} \sigma): \mathbf{N} \stackrel{\top}{\Rightarrow} \mathbf{K} \stackrel{\cdot}{\longrightarrow} \mathbf{N} \stackrel{\top}{\Rightarrow} \mathbf{K}$ at s is given, in terms of this representation, by the morphism $(\mathbf{N} \stackrel{\top}{\Rightarrow} \mathbf{H})_s: (s \multimap \mathbf{H}) \to (s \multimap \mathbf{K})$, which can be described by $\lambda h.\sigma_s \circ h$.

A representation for the domain $(\mathbf{N} \Rightarrow \mathbf{H})_s$ was also briefly touched upon in Section 2. This takes the form of the collection of functions $\mathcal{F}_s(\mathbf{H})$ with domain $\mathcal{N}$. We can also describe the actions of natural transformations in terms of these representations. If $\sigma: \mathbf{H} \stackrel{\cdot}{\longrightarrow} \mathbf{K}$ then the action of the natural transformation $(id \Rightarrow \sigma): (\mathbf{N} \Rightarrow \mathbf{H}) \stackrel{\cdot}{\longrightarrow} (\mathbf{N} \Rightarrow \mathbf{K})$ at s is given by the morphism $(id \Rightarrow \sigma)_s: \mathcal{F}_s(\mathbf{H}) \to \mathcal{F}_s(\mathbf{K})$, described by $\lambda f.\lambda n.(\sigma_{s \cup \{n\}} \circ f(n))$.

The predomain $\mathbf{C}$ is isomorphic to $(\mathbf{N} \otimes \mathbf{P}) + \mathbf{s}(\mathbf{P})$ and it is also convenient to develop a concrete representation for the predomains $\mathbf{C}_s$. We know that $(\mathbf{N} \otimes \mathbf{H})_s$ can be represented by $(s \multimap^{ne} \mathbf{H}_s)$ and $\mathbf{s}(\mathbf{H})_s$ is determined by an a element of $\mathbf{H}_{s \cup \{y\}}$ for an arbitrary y not in s. Combining both of these we obtain a representation of $((\mathbf{N} \otimes \mathbf{H}) + \mathbf{s}(\mathbf{H}))_s$ as a partial function with domain $\mathcal{N}$. Let $\mathcal{PF}_s(\mathbf{H})$ be the set of all *non-empty* partial functions f with domain a subset of $\mathcal{N}$ which satisfies

- $f(n) \in \mathbf{H}_{s \cup \{n\}}$
- $domain(f) - s$ is either $\emptyset$ or $\mathcal{N} - s$
- for all n, $m \in domain(f) - s$, $f(n) = \mathbf{H}_{(id + m \mapsto n)} f(m)$.

Functions in $\mathcal{PF}_s(\mathbf{H})$ can be ordered in the standard fashion, for partial functions: $f \leq g$ if

- $domain(g) \leq domain(f)$
- for all $n \in domain(g)$, $f(n) \leq g(n)$.

Under this ordering $\mathcal{PF}_s(\mathbf{H})$ is a predomain isomorphic to $((\mathbf{N} \otimes \mathbf{H}) + \mathbf{s}(\mathbf{H}))_s$. Moreover the actions of natural transformations can also be described in terms of this representation. Let $\sigma: \mathbf{H} \stackrel{\cdot}{\longrightarrow} \mathbf{K}$. Then the action of the natural transformation

$$(id \otimes \sigma + \mathbf{s}(\sigma)): (\mathbf{N} \otimes \mathbf{H} + \mathbf{s}(\mathbf{H})) \stackrel{\cdot}{\longrightarrow} (\mathbf{N} \otimes \mathbf{K} + \mathbf{s}(\mathbf{K}))$$

at s is given by the morphism $(id \otimes \sigma + \mathbf{s}(\sigma))_s: \mathcal{PF}_s(\mathbf{H}) \to \mathcal{PF}_s(\mathbf{K})$ described by $\lambda f.\lambda n \in domain(f).(\sigma_{s \cup \{n\}} \circ f)n$.

These representations will be useful in reasoning about elements of $\mathbf{P}$. In particular they can be used to develop a convenient notation for elements of $\mathbf{P}_s$. Let d be an element of $\mathbf{P}_s$ which is different from $\perp$. Then, modulo the isomorphisms $unfold$ and $fold$,

1. let $\mathcal{A}_s(d)$ denote $(\pi_1 \circ down)d$; $\mathcal{A}_s(d)$ is an acceptance set over $s + s$. Using the convention that $in_l(n)$, $in_r(n)$ represent $n?$, $n!$ respectively this can be taken to be an acceptance set over Ind.

2. let $d^?$ denote $(\pi_2 \circ down)d$; using the representation given above $d^?$ is an element of $(s \rightharpoonup \mathbf{F}_s)$.

3. let $d^!$ denote $(\pi_3 \circ down)d$ which also can be taken to be an element of $(s \rightharpoonup \mathbf{C}_s)$.

With this notation we can now associate *acceptance* with elements of $\mathbf{P}_s$. For any $d \in \mathbf{P}_s$ let

1. $d \models_s \Omega$ if $unfold(d) = \bot$

2. $d \models_s \mathtt{A}$ if $\mathtt{A} \in \mathtt{Acc}_s(d)$

3. $d \models_s \mathtt{n?x.a}$ if $d^? \, n \, x \models_{s \cup \{x\}} \mathtt{a}$

4. for $x \in s$, $d \models_s \mathtt{n!x.a}$ if $d^! \, n \, x \models_s \mathtt{a}$

5. $d \models_s \nu(\mathtt{y}) \, . \, \mathtt{n!y.a}$ if for some $z \notin s$, $d! \, n \, z \models_{s \cup \{z\}} \mathtt{a}[z/y]$

The uniformity of the family of domains $\mathbf{P}_s$ means that satisfiability of acceptances can be transferred from one to the other. Any $s \stackrel{\sigma}{\longmapsto} s'$ can also be viewed as a substitution over names; it leaves untouched all those which do not appear in its domain s. So, viewing acceptances as simple terms, this means that these injections can also be applied to acceptances. With this notation we have

Proposition 5.5 *If* $s \stackrel{\sigma}{\longmapsto} s'$ *then* $d \models_s \mathtt{a}$ *implie* $\mathbf{P}_\sigma(d) \models^{s'} \mathtt{a}\sigma$.

Proof: By induction on the structure of $\mathtt{a}$, by analysing the action of $\mathbf{P}_\sigma$ on the representations of the components of $\mathbf{P}$ given above. □

The most significant properties of these acceptances are given in the following lemma:

Lemma 5.6

1. $d \models_s \mathtt{n?x.a}$ *if and only if* $d^? nx \models_{s \cup \{x\}} \mathtt{a}$

2. *for* $x \in s$, $d \models_s \mathtt{n!x.a}$ *if and only if* $d^! nx \models_s \mathtt{a}$

3. $d \models_s \nu(\mathtt{y}) \, . \, \mathtt{n!y.a}$ *if and only if for every* $m \notin s$, $d!nm \models_{s \cup \{m\}} \mathtt{a[m/y]}$

Proof: The first two statements follow directly from the definitions. The third follows from the previous Proposition, using the uniformity of the functions in $\mathcal{PF}_s(\mathbf{P})$; in particular the fact that $f(n) = \mathbf{P}_{(s+m \mapsto n)} f(m)$. □

Let $\mathtt{Acc}_s(d)$ denote the set $\{ \mathtt{a} \mid d \models_s \mathtt{a} \}$.

Proposition 5.7 *For all* $d, e \in \mathbf{P}_s$, $d \leq e$ *implie* $\mathtt{Acc}_s(e) \ll \mathtt{Acc}_s(d)$.

Proof: It is straightforward to show by induction on $\mathtt{a}$ that $\forall\, s \forall\, d,\ e \in \mathbf{P}_s,\ d \leq e,\ e \models_s \mathtt{a}$ implies $d \models_s \mathtt{a}'$ for some $\mathtt{a}' \ll \mathtt{a}$. □

The converse however is less straightforward as it depends on the fact that $\mathbf{P}$ is the initial solution to it's defining equation. We refer the reader to [9] for the general underlying theory but here we simply state the relevant characteristics of $\mathbf{P}$. We define three families of natural transformations

$$\begin{aligned} \mathbf{id}^k &: \mathbf{P} \stackrel{\cdot}{\longrightarrow} \mathbf{P} \\ \mathbf{fid}^k &: \mathbf{F} \longrightarrow \mathbf{F} \\ \mathbf{cid}^k &: \mathbf{C} \stackrel{\cdot}{\longrightarrow} \mathbf{C} \end{aligned}$$

as follows:

1. $\mathbf{aid}^0 = \bot$, i.e. for $\mathbf{A} = \mathbf{P}, \mathbf{F}, \mathbf{C}$ $\mathbf{aid}^0$ is the natural transformation whose action at every s is given by the constant morphism $\lambda x.\bot$
2. $\mathbf{fid}^{k+1} = (id \Rightarrow \mathbf{id}^k)$
3. $\mathbf{cid}^{k+1} = (id \quad \mathbf{id}^k) + \mathbf{s}(\ \mathbf{id}^k)$
4. $\mathbf{id}^{k+1} = (id \times (id \quad {}^{\top}\ \mathbf{fid}^{k+1}) \times (id \quad {}^{\top}\ \mathbf{cid}^{k+1}))_{\bot}$

We state without proof:

Theorem 5.8 *For* $\mathbf{A} = \mathbf{P}, \mathbf{C}, \mathbf{F}$, $\bigvee_k(\mathbf{aid}^k) = id_{\mathbf{A}}$. □

With this characterisation of the domains we can now prove

Pro osition 5.9 *For all* $d, e \in \mathbf{P}_s$, $\mathtt{Acc}_s(e) \ll \mathtt{Acc}_s(d)$ *implie* $d \leq e$.

Proof: From the previous Theorem it is sufficient to show that for all k, $\mathbf{id}_s^k(d) \leq \mathbf{id}_s^k(e)$. The case $k = 0$ is trivial and we prove the case $k+1$ under the assumption that it is true for k.

We may assume $d \neq \bot$ and since $\mathtt{Acc}(e) \ll \mathtt{Acc}(d)$ this also means that $e \neq \bot$. Using the representations developed above this means that $down(\ \mathbf{id}_s^{k+1}(d))$ and $down(\ \mathbf{id}_s^{k+1}(e))$ may be taken to be

$$\mathcal{A}_s(d) \times (\mathbf{fid}_s^{k+1} \circ d^?) \times (\mathbf{cid}_s^{k+1} \circ d^!) \text{ and } \mathcal{A}_s(e) \times (\mathbf{fid}_s^{k+1} \circ e^?) \times (\mathbf{cid}_s^{k+1} \circ e^!)$$

respectively. Now $\mathtt{Acc}_s(e) \ll \mathtt{Acc}_s(d)$ means that $\cup\mathcal{A}_s(e) \subseteq \cup\mathcal{A}_s(d)$ and therefore $\mathcal{A}_s(e) \subseteq \mathcal{A}_s(d)$, since the latter are acceptance sets. It remains to prove

$$\begin{aligned} \mathbf{fid}_s^{k+1} \circ d^? &\leq \mathbf{fid}_s^{k+1} \circ e^? \\ \mathbf{cid}_s^{k+1} \circ d^! &\leq \mathbf{cid}_s^{k+1} \circ d^! \end{aligned}$$

As an example we prove the latter. Both $\mathbf{cid}_s^{k+1} \circ d^!$ and $\mathbf{cid}_s^{k+1} \circ e^!$ are partial functions, using the representations above, and so we first must demonstrate that the domain of $\mathbf{cid}_s^{k+1} \circ e^!$ is contained in that of $\mathbf{cid}_s^{k+1} \circ d^!$. However this follows immediately from the fact that $\mathtt{Acc}_s(e) \ll \mathtt{Acc}_s(d)$ since $n \in domain(e^!)$ if and only if $e \models_s \mathtt{n!x.a}$ for some $\mathtt{x}$, $\mathtt{a}$.

So suppose $n \in domain(e!)$. We must show that

$$(\mathbf{cid}_s^{k+1} \circ d^!)n \leq (\mathbf{cid}_s^{k+1} \circ e^!)n$$

as objects in $\mathbf{C}_s$, i.e $\mathcal{PF}_s(\mathbf{P})$. Using the representation of the action of functors described above $(\mathbf{cid}_s^{k+1} \circ d^!)n$, $(\mathbf{cid}_s^{k+1} \circ e^!)n$ work out to be

$$\lambda m \in domain(d^!n).\ \mathbf{id}^k(d^!nm) \text{ and } \lambda m \in domain(e^!n).\ \mathbf{id}^k(e^!nm)$$

respectively. So we must prove

- $domain(e^!n) \subseteq domain(d^!n)$.

  For any $m \in s$, $m \in domain(e^!n)$ if and only if e |

$$
\begin{array}{rcl}
X \oplus X & = & X \\
X \oplus Y & = & Y + X \\
(X \oplus Y) \oplus Z & = & X \oplus (Y \oplus Z) \\
X \oplus Y & \leq & X \\
X + X & = & X \\
X + Y & = & Y + X \\
(X + Y) + Z & = & X + (Y + Z) \\
X + 0 & = & X \\
X + (Y \oplus Z) & = & (X + Y) \oplus (X + Z) \\
X \oplus (Y + Z) & = & (X \oplus Y) + (X \oplus Z)
\end{array}
$$

$$
\begin{array}{rcl}
n\,?\,\lambda\,x.\,X + n\,?\,\lambda\,x.\,Y & = & n\,?\,\lambda\,x.\,(X \oplus Y) \\
n\,!\,x\,.\,X + n\,!\,x\,.\,Y & = & n\,!\,x\,.\,(X \oplus Y) \\
n\,?\,\lambda\,x.\,X + n\,?\,\lambda\,x.\,Y & = & n\,?\,\lambda\,x.\,X \oplus n\,?\,\lambda\,x.\,Y \\
n\,!\,x\,.\,X + n\,!\,x'\,.\,Y & = & n\,!\,x\,.\,X \oplus n\,!\,x'\,.\,Y
\end{array}
$$

$$
\begin{array}{rcl}
\nu(n)\,.\,(X + Y) & = & \nu(n)\,.\,X \;+\; \nu(n)\,.\,Y \\
\nu(n)\,.\,(X \oplus Y) & = & \nu(n)\,.\,X \;\oplus\; \nu(n)\,.\,Y \\
\nu(n)\,.\,n\,?\,X & = & 0 \\
\nu(n)\,.\,n\,!\,x\,.\,X & = & 0 \\
\nu(n)\,.\,m\,?\,\lambda\,y.\,X & = & m\,?\,\lambda\,y.\,\nu(n)\,.\,X \;\text{ if } n \neq y,\; n \neq m \\
\nu(n)\,.\,m\,!\,y\,.\,X & = & m\,!\,y\,.\,\nu(n)\,.\,X \;\text{ if } n \neq y, n \neq m \\
\nu(n)\,.\,\nu(m)\,.\,X & = & \nu(m)\,.\,\nu(n)\,.\,X
\end{array}
$$

$$
\begin{array}{rcl}
(X \oplus Y) \parallel Z & = & (X \parallel Z) \oplus (Y \parallel Z) \\
X \parallel Y & = & Y \parallel X
\end{array}
$$

$$
\begin{array}{rcl}
X + \Omega & \leq & \Omega \\
X \parallel (Y + \Omega) & = & \Omega \\
\Omega & \leq & X \\
\nu(n)\,.\,\Omega & = & \Omega
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{rec}\,\mathsf{X}\,.\,t & = & t[\mathsf{rec}\,\mathsf{X}\,.\,t/X]
\end{array}
$$

Figure 3: Equations

Let p, q denote $\sum\{\, pre_i p_i \mid i \in I \,\}$, $\sum\{\, pre_j q_j \mid j \in J \,\}$, where each pre_k has the form $n!x$ or $n?\lambda y.$, and assume that every bound variable is different from the free variables in p, q. Then

$$p \parallel q = \begin{cases} ext(p,q) & \text{if } comm\ (p,q) = fal\ e \\ (ext(p,q) + int(p,q)) \oplus int(p,q) & \text{otherwise} \end{cases}$$

where

$$ext(p,q) = (\mathrm{p}; \mathrm{q}$$

I $\quad \dfrac{}{p \leq p} \qquad \dfrac{p \leq q,\ q \leq r}{p \leq r}$

II $\quad \dfrac{p_i \leq q_i}{op(p_1 \ldots p_n) \leq op(q_1 \ldots q_n)}$

$op = +, \oplus, \|, \nu(n).XS$

Eq $\quad \dfrac{}{p \leq q}$

for every instance of an inequation

$\alpha \quad \dfrac{p \equiv_\alpha q}{p = q}$

Input $\quad \dfrac{p[z/x] \leq q[z/x] \text{ for all names } z}{n\,?\,\lambda x.p \leq n\,?\,\lambda x.q}$

If $\quad \dfrac{p \leq q}{\mathsf{if}\ b\ \mathsf{then}\ p\ \mathsf{else}\ p' \leq \mathsf{if}\ b\ \mathsf{then}\ q\ \mathsf{else}\ q'} \qquad \dfrac{[\![b]\!] = true}{\mathsf{if}\ b\ \mathsf{then}\ X\ \mathsf{else}\ Y = X}$

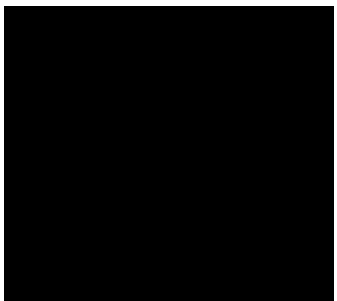
$\dfrac{p' \leq q'}{\mathsf{if}\ b\ \mathsf{then}\ p\ \mathsf{else}\ p' \leq \mathsf{if}\ b\ \mathsf{then}\ q\ \mathsf{else}\ q'} \qquad \dfrac{[\![b]\!] = fal\ e}{\mathsf{if}\ b\ \mathsf{then}\ X\ \mathsf{else}\ Y = Y}$

Figure 5: The Proof System

Lemma 5.13 *If $\underline{x} \vdash h : P$, s contain all x_i and id i any environment in ENV_s which i the identity on s then $p \Downarrow$ if and only if $[\![p]\!]_s id \neq \bot$.*

Proof: (*Outline*) First suppose $p \Downarrow$. In this case we know that p has a *hnf* h and an examination of the definitions of $+_{\mathbf{P}}$, $\oplus_{\mathbf{P}}$ shows that $[\![p]\!]_s id$ is different from $\bot$.

The converse depends on the remark that for any *recur ion free* process term d, if $d \Uparrow$ then $[\![d]\!]_s \bot = \bot$. This can be proven as follows: one can show by structural induction on d that $d \Uparrow$ implies $\vdash d = \Omega$ and since $[\![\Omega]\!] = \bot$ and the proof system is sound for the interpretation $[\![\ ]\!]_s id$ this means $[\![d]\!] id = \bot$.

Now 

is sound with respect to $\ll$ it follows that $p \Downarrow$. $\square$

The syntactic structure of *hnf* ensure that we can determine the various components of their denotations. This is the topic of the next three Lemmas where we assume h is a *hnf* of the form described in Definition 2 above, such that $\underline{x} \vdash h : P$ and $x_i \in s$ for every x_i in $\underline{x}$. For any $\sigma \in ENV_s$, $[\![h]\!]_s\sigma \in \mathbf{P}_s$ and we will see how the components of $[\![h]\!]_s\sigma$ are determined by the syntax of h. For simplicity we assume that σ is an injection.

Lemma 5.14 $\mathtt{Acc}_s([\![h]\!]_s\sigma) = \sigma(\mathcal{A})$

Proof: An examination of the definition of $+_{\mathbf{P}}$ and $\oplus_{\mathbf{P}}$ as it applies to *hnf* . $\square$

As an object in $\mathbf{P}_s$, $[\![h]\!]_s\sigma$ is different from $\perp$ and therefore using the notation developed in Section 5.1 $([\![h]\!]_s\sigma)^?$ can be considered as an element in $s \rightharpoonup \mathbf{F}_s$. For any $m? \in \cup\mathcal{A}$, $h^{m?}$ is an abstraction term and therefore $[\![h^{m?}]\!]_s\sigma \in \mathbf{F}_s$. If $\sigma(m) = n$ then

Lemma 5.15 $([\![h]\!]_s\sigma)^?n = [\![h^{m?}]\!]_s\sigma$

Proof: Again a simple examination of the interpretation of *hnf* . $\square$

In a similar manner $([\![h]\!]_s\sigma)^!$ can be considered as an element of $s \rightharpoonup \mathbf{C}_s$ and $([\![h]\!]_s\sigma)^!n$, a function in $\mathcal{PF}_s(\mathbf{P}_s)$, can be defined using $h^{m!}$. Suppose this has the form

$$m\,!\,y_1\,.\,h_1 + \ldots + m\,!\,y_k\,.\,h_k + \nu(y)\,.\,m\,!\,y\,.\,h_y.$$

Let $sem_{m!}(h) \in \mathcal{PF}(\mathbf{P}_s$

1. Suppose $p \models^b \mathtt{n?y.b}$. Let $p^{n?}$ have the form $\lambda x.t$. This means that $t[y/x] \models^b \mathtt{b}$. Applying induction we have

$$[\![t[y/x]]\!]_{s\cup\{y\}} id \models_{s\cup\{y\}} \mathtt{b'}$$

for some $\mathtt{b'}$ such that $\mathtt{b'} \ll \mathtt{b}$. By the *Sub titution Lemma* this means

$$[\![t]\!]_{s\cup\{y\}}(id[x \mapsto y]) \models_{s\cup\{y\}} \mathtt{b'}.$$

However if we calculate $([\![p]\!]_s id)^? n$, as an element of $\mathcal{F}(\mathbf{P}_s)$, using Lemma 5.15, we obtain the function

$$\lambda k \in \mathcal{N}.[\![t]\!]_{s\cup\{k\}}(id[x \mapsto k])$$

and so by definition $[\![p]\!]_s id \models_s \mathtt{n?y.b'}$.

All of these steps are reversible and therefore the converse also holds.

2. Suppose $[\![p]\!]_s id \models_s \nu(\mathtt{m})\,.\,\mathtt{n!m.b}$ and for convenience let us assume that $m \in s$. Using Lemma 5.16 this means that for all $z \notin s$

$$sem_{n!}(p)z \models_{s\cup\{z\}} \mathtt{b[z/m]}$$

In particular choosing z to be m we have

$$[\![p_y]\!]_{s\cup\{m\}}(id[y \mapsto m]) \models_{s\cup\{m\}} \mathtt{b}.$$

Using the *Sub titution Lemma* this means

$$[\![p_y[m/y]]\!]_{s\cup\{m\}} id \models_{s\cup\{m\}} \mathtt{b}.$$

By induction there exists some $\mathtt{b'} \ll \mathtt{b}$ such that $p_y[m/y] \models^b \mathtt{b'}$ and since $p \stackrel{\nu(m)\,.\,n!m}{\Longrightarrow} p_y[m/y]$ it follows that $p \models \nu(\mathtt{m})\,.\,\mathtt{n!m.b'}$.

Again these steps are reversible and thus the converse also holds.

□

We have used the proof system developed here only to reduce process terms to head normal forms. However it is also possible to show that it is complete with respect to the model, for recursion free terms. Specifically we can show, for any recursion free process term d and any process term q, that $\vdash d \leq q$ if and only $[\![d]\!]_s id \leq [\![q]\!]_s id$, where s is any set including the free names occurring in d, q.

5.3 Behavioural Characterisation

The behavioural characterisation of process terms using *acceptance* depends on a detailed analysis of the behavioural semantics. This was originally given in terms of a reduction relation whose definition uses a structural equivalence and consequently it is often not straightforward to derive it properties. However the following can be proved by proof induction on the definition of $\stackrel{\tau}{\longrightarrow}$:

Lemma 5.18 *$p \xrightarrow{\tau} q$ implie $p\sigma \xrightarrow{\tau} q\sigma$ for every ub titution σ.*

The following lemma will assist with the behavioural analysis.

Lemma 5.19 *$p \parallel q \xrightarrow{\tau} r$ if and only if one of the following:*

1. *$r \equiv p' \parallel q$ where $p \xrightarrow{\tau} p'$*
2. *$r \equiv p \parallel q'$ where $q \xrightarrow{\tau} q'$*
3. *$r \equiv p' \parallel q'$ where $p \xrightarrow{n?y} p'$ and $q \xrightarrow{n!y} q'$*
4. *$r \equiv (\nu(z)\,.\,p' \parallel q')$ for all $z \notin fv(p,q)$ and*

$$p \xrightarrow{x?z} p',\ q \xrightarrow{\nu(z)\,.\,n!z} q' \quad \text{or} \quad q \xrightarrow{x?z} q',\ p \xrightarrow{\nu(z)\,.\,n!z} p'$$

□

For each acceptance set $\mathtt{a}$ we design a specific test $t(\mathtt{a})$ with the property that

$$\forall \mathtt{a}' \ll \mathtt{a}.p \not\models^b \mathtt{a}' \quad \text{implies} \quad p\ mu\ t\ t(a).$$

The definition of the these tests actually requires two extra parameters, one a finite set of names, the other a finite subset of *Ind*.

1. $\mathtt{a} = \Omega$
   $t(\mathtt{a})^B_X = 1.\omega$ (an abbreviation for $\omega \oplus \omega$)
2. $\mathtt{a} = \mathtt{A}$
   $t(\mathtt{a})^B_X = \sum\{\, n\,!\,y\,.\,\omega \mid n! \in B \,\} + \sum\{\, n\,?\,\lambda y.\,\omega \mid n? \in B \,\}$
3. $\mathtt{a} = \mathtt{n?y.b}$
   $t(\mathtt{a})^B_X = 1.\omega + n\,!\,y\,.\,t(\mathtt{b}^B_{X\cup\{y\}})$
4. $\mathtt{a} = \mathtt{n!y.b}$
   $t(\mathtt{a})^B_X = 1.\omega + n\,?\,\lambda z.\,\mathsf{if}\ z = y\ \mathsf{then}\ t(\mathtt{b}^B_X\ \mathsf{else}\ \omega)$, where $z \notin fv(t(\mathtt{b})^B_{X\cup\{y\}}, y))$
5. $\mathtt{a} = \nu(\mathtt{y})\,.\,\mathtt{n!y.b}$
   $t(\mathtt{a})^B_X = 1.\omega + n\,?\,\lambda z.\,\mathsf{if}\ z \in X\ \mathsf{then}\ w\ \mathsf{else}\ t(\mathtt{b}^B_{X\cup\{y\}})$, where $z \notin fv(t(\mathtt{b})^B_{X\cup\{y\}}, y))$ and

- $A(\texttt{Omega}) = \emptyset$
- $A(\texttt{B}) = \texttt{B}$
- $A(\alpha.\texttt{b}) = A(\texttt{b})$

Proposition 5.21 *If $fv(p) \subseteq X$ then ($\forall a'$ such that $a' \ll a$, $p \not\models^b a'$) if and only if there exists some B such that $B \cap A(\texttt{a}) = \emptyset$ and p must $t(a)^B_X$.*

- $\mathtt{a} = \nu(y)\,.\,n?$

- $n!x$, $n?x$
- $n?x$, $\nu(x)\,.\,n!x$
- $\nu(x)\,.\,n!x$, $n?x$

for some x. Moreover we can assume that in the last two cases the particular bound name x is chosen so that it is does not occur in $fv(p)$. Further, because $r_k \not\xrightarrow{\tau}$ we also know that $S(q_m) \cap \overline{S(e_m)} = \emptyset$, where $\overline{S} = \{\, n? \mid n! \in S \,\} \cup \{\, n! \mid n? \in S \,\}$. For convenience let s denote the sequence of actions a_i, S the set of indications $S(q_m)$ and $\mathtt{a}$ the *acceptance* $\mathtt{sS}$. Since $p \ll q$ this means that $p \models^b \mathtt{b}$ for some $\mathtt{b} \ll \mathtt{a}$.

There are now two cases to consider:

1. $\mathtt{b}$ has the form $\mathtt{sT}$. By definition $p \stackrel{s}{\Longrightarrow} p'$ for some p' such that $p' \not\xrightarrow{\tau}$ and $S(p') \subseteq T \subseteq S$. This computation, from p to p' can be zipped together with $e \stackrel{\overline{s}}{\Longrightarrow} e_m$ to form a computation

   $$e \parallel \xrightarrow{\tau} \ldots \xrightarrow{\tau} e_m \parallel p'$$

   The relevant features of this computation are firstly that $e_m \parallel p' \not\xrightarrow{\tau}$ and secondly it only uses e_i which occur in the original computation $(*)$. It follows from $p\ mu\ t\ e$ that $e_i \xrightarrow{\omega}$ for some i.

2. $\mathtt{b}$ has the form $\mathtt{s'\Omega}$ where s' is some subsequence of s. The approach of the previous case also works here using the computation $p \stackrel{s'}{\Longrightarrow} p'$ where $p \Uparrow$.

We have not touched on the case when the computation $(*)$ is infinite. However this can be treated in exactly the same manner as in Lemma 4.4.13 from [10]; we simply need to know that, up to α

least element. Moreover because of the degeneracy of much of the algebraic structure of upper

$\tau\colon (A \times B) \stackrel{\cdot}{\longrightarrow} C$ can be extended in an obvious way to a natural transformation in $(\mathbf{N}^{\top} A \times B) \stackrel{\cdot}{\longrightarrow} \mathbf{N}^{\top} C$ and let us use the non-standard notation $\mathbf{N}^{\top} \tau$ for this extension.

Then the natural transformation $\mathbf{ext}\colon (\mathbf{P} \times \mathbf{P}) \stackrel{\cdot}{\longrightarrow} \mathbf{P}$ is given by $(\mathbf{ar_A} \times \mathbf{ar_F} \times$

References

[1] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.

[2] S. Abramsky and C. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 1989. to appear.

[3] Henk Barendregt. *The Lambda Calculu* . North-Holland, 1984. Studies in logic 103.

[4] M. Boreale and R. DeNicola. Testing for mobile processes. In *Proceeding of CONCUR 92*, 1992. To appear in *Information and Computation.*

[5] G. Boudol. A lambda calculus for (strict) parallel functions. *Information and Computation*, 108:51–127, 1994.

[6] R. Crole. *Categorie for Type* . Cambridge University Press, 1993.

[7] R. DeNicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 24:83–113, 1984.

[8] F.J.Oles. Type algebras, functor categories and block structure. In M. Nivat and J. Reynolds, editors, *Algebraic Method in Semantic* , pages 543–573. Cambridge University Press, 1985.

[9] Carl Gunter. *Semantic of Programming Language* . MIT Press, 1992.

[10] M. Hennessy. *An Algebraic Theory of Proce e* . MIT Press, 1988.

[11] M. Hennessy. A Model for the π Calculus. Technical Report 8/91, University of Sussex, 1991.

[12] M. Hennessy and A. Ingolfsdottir. A theory of communicating processes with value-passing. *Information and Computation*, 107(2):202–236, 1993.

[13] D.J. Howe. Equality in lazy computation systems. In *Proceeding of the 4th IEEE Sympo ium on Logic in Computer Science*, pages 198–203, 1989.

[14] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[15] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1):1–40, 1992.

[16] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part ii. *Information and Computation*, 100(1):41–77, 1992.

[17] Robin Milner. The polyadic π-calculus: a tutorial. In *Proc. International Summer School on Logic and Algebra of Specification*, Marktoberdorf, 1991.

[18] Eugenio Moggi. Notions of computation and monad. *Information and Computation*, 93:55–92, 1991.

[19] E. Moggi, M.P.Fiore and D. Sangiorgi. A Fully-Abstract Model for the π-Calculus. To appear in Proceedings of LICS'96, 1996.

[20] P.W. O'Hearn and R.D. Tennant. Semantics of local variables. In *Application of Cate1Tdgiorgi.*