ELSEVIER

Theoretical Computer Science 278 (2002) 53–89

Theoretical Computer Science

www.elsevier.com/locate/tcs

A fully abstract denotational semantics for the π-calculus ☆

Matthew Hennessy

School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton BN1 9QH, UK

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.

1. Introduction

The π-calculus [15, 16] is a process algebra for describing processes which communicate by exchanging *channel names*. 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-passing* process algebras.

Numerous semantic theories have been proposed for the π-calculus, and it variants, [4, 16, 21, 24]. 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 *testing 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.

E-mail address: matthewh@cogs.susx.ac.uk (M. Hennessy).

0304-3975/02/$ - see front matter
PII: S0304-3975(00)00331-5

The use of *private* names has made it very difficult to develop set-theoretic denotational models for the π-calculus. Their behavioural properties have much in common, at least intuitively, with that of local variables in Algol-like languages and references in ML-like languages. In papers, such as [8, 20, 22], categorical methods have been developed for providing denotational models for languages with such constructs, and recently, in [19, 25], these techniques have been applied to provide set-theoretic denotational models for fragments of the π-calculus, which are *fully abstract* with respect to *strong bisimulation equivalence*. The objective of this present paper is to show that similar techniques can be employed to provide set-theoretic models for the π-calculus which generalise those presented in [12] for value-passing process calculi. In particular, we provide two set-theoretic denotational models for the π-calculus which are *fully abstract* with respect to *must* and *may* testing, respectively.

The π-calculus consists of a small number of *constructors* for defining processes. For example $n?\lambda x.t$ describes a process which inputs some name m, from a pre-defined set of names $\mathcal{N}$, along the communication channel named n and behaves according to the result of applying the abstraction $\lambda x.t$ to the name m. Thus if $\mathbf{P}$ is to act as a model for the π-calculus then to interpret this input construct we require a function $\mathbf{in_P}$ of type $(\mathbf{N} \times (\mathbf{N} \Rightarrow \mathbf{P})) \Rightarrow \mathbf{P}$, where $\mathbf{N}$ is some appropriate representation of the set of names $\mathcal{N}$ and we assume abstractions are interpreted in $(\mathbf{N} \Rightarrow \mathbf{P})$. Similarly, the term $n!m.t$ represents a process which can output the name m on the channel n and then behaves as t. To interpret this construct we require a function $\mathbf{out_P}$ of type $(\mathbf{N} \times \mathbf{N} \times \mathbf{P}) \Rightarrow \mathbf{P}$.

In addition to these input/output constructors there are also binary combinators such as $t \| u$, to run the processes t and u in parallel, and $t + u$ to choose between t and u; these require functions $\|_{\mathbf{P}}$, $+_{\mathbf{P}}$ of type $(\mathbf{P} \times \mathbf{P}) \Rightarrow \mathbf{P}$. However the most intriguing constructor is *restriction*, $\nu(n)t$. For example, the process $\nu(n)m!n.t$ can output along the channel m, to a parallel process, some new private name k, different from all others in existence, and this new name k may be used for subsequent communication between the parallel process and $t[k/n]$. Since $\nu(n)$ is a *binding* operator it needs to be interpreted as an operator of type $(\mathbf{N} \Rightarrow \mathbf{P}) \Rightarrow \mathbf{P}$.

We now discuss briefly the nature of the model $\mathbf{P}$ and how it can support the interpretation of these constructors. Following the ideas of [25], there are two important observations which underlie the nature of the model:

(1) The behaviour of a process depends only on the *finite* set of channel names which it contains, or to be more precise the set of its *free* channel names, i.e. those which are not bound by the binding constructors of input and restriction. So for each finite set of names s we will have a model $\mathbf{P}_s$ to interpret the behaviour of process terms which only use the names in s.

(2) Intuitively, the behaviour of a process is preserved by injective renamings of its (free) names. For example, let p be the simple process $n!m.0 + m!n.0$ and σ the injective renaming which maps m to k and n to l. Then, intuitively, the behaviour of $p\sigma$, i.e. $l!k.0 + k!l.0$, is determined by that of p; it can be reconstructed from that of p by means of the injection σ. More generally this implies the existence,

for each injection $s \overset{\sigma}{\mapsto} s'$, of a *translation* $\mathbf{P}_\sigma$ from the space of meanings $\mathbf{P}_s$ to that of $\mathbf{P}_{s'}$.

The most convenient framework in which to formalise this uniform family of interpretations $\{\mathbf{P}_s \mid s \subseteq_{fin} \mathcal{N}\}$ is in terms of functor categories. Let I be the category whose objects are finite subsets of $\mathcal{N}$ and whose morphisms are injections $s \overset{\sigma}{\mapsto} s'$. Let $\mathcal{D}$ be some suitable category of *domains*. 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 $s \overset{\sigma}{\mapsto} s'$. 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. For example to describe the behaviour of the process $\nu(x)n!x.t$, whose free variables are included in s, one needs to describe what happens when *some* name y not in s is output on n and this name is subsequently used to interact with the process $t[y/x]$. Behaviourally, it does not matter which y not in s we choose, and denotationally this is reflected in the fact that for any $y, y' \notin s$, $\mathbf{P}_{s\cup\{y\}}$ is isomorphic to $\mathbf{P}_{s\cup\{y'\}}$: the isomorphism is given by the pair of morphisms

$$\mathbf{P}_{id+y\mapsto y'} : \mathbf{P}_{s\cup\{y\}} \to \mathbf{P}_{s\cup\{y'\}} \quad \text{and} \quad \mathbf{P}_{id+y'\mapsto y} : \mathbf{P}_{s\cup\{y'\}} \to \mathbf{P}_{s\cup\{y\}}.$$

Formally, I is a symmetric monoidal category with disjoint union being the tensor, which we denote by $+$. We use 1 to denote any singleton set, since all such sets are isomorphic. To reflect this behaviour the domain equation for $\mathbf{P}$ will have an output component which includes $\mathbf{N} \times \mathbf{s}(\mathbf{P})$ where $\mathbf{s}$ is the functor defined so that the action of $\mathbf{s}(A)$ at s, for any functor A, gives A_{s+1}. Intuitively this component, at s, models the output along a channel of an *arbitrary* new name.

Before giving the actual domain equation for $\mathbf{P}$ we need to first discuss the category of domains $\mathcal{D}$ used. Since our version of the π-calculus includes recursively defined processes we use *algebraic cpos*, [9]. In addition the language has operators for internal and external choice, $\otimes$ and $+$, respectively. With respect to the behavioural testing preorders [10] these operators enjoy many algebraic properties. These are given in

Section 2 and a domain which satisfies these laws is called a *choice domain*. For example with respect to the *must* preorder $\oplus$ satisfies the laws of a lower semi-lattice. Consequently to model this behavioural preorder we work within the category $l\mathcal{S}\mathcal{L}^I$ where $l\mathcal{S}\mathcal{L}$ is the category whose objects are domains which are also lower semi-lattices and whose morphisms are linear continuous functions, i.e. continuous functions which also preserve the semi-lattice meet operator. $l\mathcal{S}\mathcal{L}^I$ has a natural exponential operator, $\Rightarrow$, which has a left-adjoint, a form of tensor product which we denote by $\otimes$. In short $l\mathcal{S}\mathcal{L}^I$ is an *autonomous* category, i.e. a closed symmetric monoidal category.

We use as our domain the initial solution to the equation

$$\mathbf{P} \cong (\mathbf{A} \times (\mathbf{N} \otimes \mathbf{F})^{\top} \times (\mathbf{N} \otimes \mathbf{C})^{\top})_{\perp}$$

$$\mathbf{F} \cong \mathbf{N} \Rightarrow \mathbf{P}$$

$$\mathbf{C} \cong (\mathbf{N} \otimes \mathbf{P}) + \mathbf{s}(\mathbf{P})$$

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

- $\mathbf{A}$, a functor representing the initial nondeterministic behaviour of a process,
- $\mathbf{N} \otimes \mathbf{F}$, a functor representing the input behaviour, as explained briefly above,
- $\mathbf{N} \otimes \mathbf{C}$ a functor representing the output behaviour. $\mathbf{C}$, representing the possibilities of output on a particular channel, has two components, $\mathbf{N} \otimes \mathbf{P}$ representing the standard output of a name, and $\mathbf{s}(\mathbf{P})$ representing the output of an *arbitrary* new name, as explained briefly above.

Since $\mathbf{P}$ is an object in $l\mathcal{S}\mathcal{L}^I$ it automatically has an appropriate interpretation for the internal choice operator $\oplus$, namely the meet operation. Moreover properties of the natural transformations used to define $\mathbf{P}$ ensure that it is also a *choice domain*, thereby providing also an interpretation for the external choice operator $+$. We show how to interpret all of the other constructors from our version of the π-calculus in $\mathbf{P}$ and prove that it is *fully abstract* with respect to the *must* testing preorder.

Essentially the same equation, only with $\top$ and $\perp$ is omitted, can be solved in the category $u\mathcal{S}\mathcal{L}^I$, where $u\mathcal{S}\mathcal{L}$ is the category of *upper* semi-lattices; the internal choice, $\oplus$, satisfies the laws of an upper semi-lattice with respect to the *may* testing preorder. We also prove that the resulting interpretation is *fully abstract* with respect to this behavioural preorder.

We now give an outline of the remainder of the paper. In Section 2 we review the various mathematical constructions needed to describe the model. The language and its operational semantics is given in Section 3. It contains the standard π-calculus constructors for input, output, restriction, parallel and the empty process, although the syntax used is somewhat different than that in [15, 16]. In addition we allow recursive definitions of processes together with name matching and mismatching; the latter takes the form of a constructor $\mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ u$ where b can be any boolean expression from a very simple language for comparing names. As discussed above we also use an internal choice operator $\oplus$, which is definable in the more standard versions of the π-calculus. Finally in place of the *CCS* choice operator, which normally forms part of the π-calculus, we use external choice. Modulo the change from *CCS* choice

to external choice the operational semantics, in the form of a reduction relation, is standard, [15]. This section ends with the definition of two behavioural preorders $\sqsubseteq_{\text{must}}$ and $\sqsubseteq_{\text{may}}$ over process terms: intuitively $p \sqsubseteq_{\text{must}} q$ ensures that every test, in the form of another process, guaranteed by p is also guaranteed by q, while $p \sqsubseteq_{\text{may}} q$ means that q is capable of satisfying all those tests which p can satisfy. The definitions are taken directly from the general framework presented in [10].

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 category $l\mathcal{S}\mathcal{L}^I$.

Section 5 contains the main result of the paper. We prove

> for all process terms p, q, $\mathbf{P}[\![p]\!] \leqslant \mathbf{P}[\![q]\!]$ if and only if $p\rho \sqsubseteq_{\text{must}} q\rho$ for every renaming ρ of the set of names $\mathcal{N}$

i.e. the model is *fully abstract* with respect to the behavioural preorder $\sqsubseteq_{\text{must}}$. 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 trace*. This consists of a finite sequence of *actions*, appropriate to the π-calculus, followed either by Ω, indicating an ability to diverge, or by a finite set of *communication potentials*, a communication potential taking the form either $n?$ or $n!$:

- The first result is an *internal full abstraction* result for the model. Sets of acceptance traces can be associated with objects in the domains $\mathbf{P}_s$, the component of $\mathbf{P}$ at the set s, and we prove that equality in the model is determined by these sets.
- Sets of acceptance traces can also be associated behaviourally with process terms of the language, using the operational semantics. The second result, in Section 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 $\sqsubseteq_{\text{must}}$ is also determined by sets of acceptance traces, those associated behaviourally with process terms.

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

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] use suitable adaptations of *bisimulation 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 version of the language is used while [11] uses essentially the same version as the present paper. Indeed much of the equational treatment of processes in Section 5.2, in particular

their reduction to head normal forms, is taken directly from this Technical Report. However, the model constructed is a *term model*, defined by quotienting process terms by the provability relation generated by the proof system and using an ideal completion construction to obtain a domain.

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 a version of the π-calculus with replication instead of recursion. 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 **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. However, here the emphasis is more on exhibiting a general framework which can be instantiated to yield semantic models for a range of languages and the use of monads as a structuring mechanism for yielding denotational semantics. Much use is made of a general meta-language within which the π-calculus can be interpreted. In particular, the proofs relating the operational and denotational semantics are expressed in terms of this meta-language.

From papers such as [8, 20, 22], it is clear that functor categories provide very useful structures within which to interpret complex operational features involving locality. The present paper, together with [19, 25], serves to emphasise that similar techniques can also play a very significant role in understanding the semantics of the π-calculus, or more generally process languages with local names.

2. Mathematical constructions

For *domains* we use algebraic cpo's in which every bounded subset has a least upper bound [9]. A *predomain* is a domain which may lack a least element, i.e. only non-empty bounded subsets have a least upper bound. However, we will generally use domains with structure. Let $l\mathscr{S}\mathscr{L}_p$ be the category whose objects are predomains D endowed with a continuous binary operator $\wedge$ which satisfy the lower semi-lattice laws:

$$x \wedge x = x$$
$$x \wedge y = y \wedge x$$
$$(x \wedge y) \wedge z = x \wedge (y \wedge z)$$
$$x \wedge y \leqslant x$$

and whose morphisms are *linear* continuous functions, i.e. continuous functions which preserve $\wedge$. This category has products and coproducts; the product is ordinary Cartesian product $D \times E$, with $\wedge$ defined pointwise; we use $\langle f, g\rangle$ to denote the unique morphism from A to $D \times E$ obtained from the morphisms $f : A \to D$, $f : A \to E$. The coproduct $D + E$ can be defined by $D^{\top} \times E^{\top} - \{\top \times \top\}$, where $D^{\top}$ is the lower semi-lattice obtained from D by adjoining a new top element $\top$. We use $[f, g]$ to denote the unique morphism from the coproduct $D + E$ determined by the morphisms f, g from D, E, respectively. The lifted domain $D_{\perp}$, obtained by adjoining new bottom element to D is also a functor in $l\mathscr{S}\mathscr{L}_p$; the meet is extended in the trivial manner by making it strict.

For any set s the set of maps from s to D, where D is an object in $l\mathscr{S}\mathscr{L}_p$, is also in $l\mathscr{S}\mathscr{L}_p$ as $\wedge$ and the order can be defined pointwise; this object is denoted by $(s \to D)$. In the same way the set of morphisms between two objects, $D_1 \Rightarrow D_2$, is also in the category. However $- \Rightarrow D_2$ is not right-adjoint to $D_1 \times -$. Nevertheless, it does have a right-adjoint, which we denote by $D_1 \otimes -$. This means that $l\mathscr{S}\mathscr{L}_p$ is naturally an autonomous category, i.e. is a closed symmetric monoidal category. The tensor product, $D_1 \otimes D_2$, can be constructed by quotienting formal finite meets of objects from the product $D_1 \times D_2$ with respect to equations dictating that $\wedge$ is linear in both arguments. Every continuous function f from $D_1 \times D_2$ to D which is linear in both arguments determines a unique morphism $f^{\otimes} : D_1 \otimes D_2 \to D$ which satisfies $f(a, b) = f^{\otimes}(a \otimes b)$. However for notational convenience we continue to use π_i, $i = 1, 2$, as the morphisms from $D_1 \otimes D_2$ to D_i determined by the projections from $D_1 \times D_2$ to D_i and $\langle f, g\rangle$ to denote the morphism into $D_1 \otimes D_2$ determined by the morphisms $f : A \to D_1$, $f : A \to D_2$. We will also use $D^{\otimes^k}$ to denote the product of k copies of D, $D \otimes \cdots \otimes D$.

Let $\hat{s}$ denote the free lower-semi-lattice generated by a finite set s; this can be represented as $\mathscr{P}^{\mathrm{ne}}(s)$, the set of non-empty subsets of s ordered by reverse subset inclusion. Since for finite domains continuity requirements degenerate to monotonicity requirements this means that $\hat{s}$ is also the free object in $l\mathscr{S}\mathscr{L}_p$ generated by the finite set s. The product $\hat{s} \otimes \cdots \otimes \hat{s}$, can be represented by $\mathscr{P}^{\mathrm{ne}}(s_1 \times \cdots \times s_k)$, again ordered by reverse subset inclusion and set union as $\wedge$. Also for any object D in $l\mathscr{S}\mathscr{L}_p$, $\hat{s} \otimes D$ has a simple representation as $(s \rightharpoonup^{\mathrm{ne}} D)$, the set of non-empty partial functions from s to D; here the order is defined by $f \leqslant g$ if

- $domain(g) \subseteq domain(f)$,
- for every $n \in domain(g)$, $f(n) \leqslant g(n)$.

and $f \wedge g$ is given by

$$\begin{aligned}(f \wedge g)x &= f(x) \wedge g(x),\ x \in domain(f) \cap domain(g)\\ &= f(x),\ x \in domain(f) - domain(g)\\ &= g(x),\ x \in domain(g) - domain(f).\end{aligned}$$

Note that this implies $domain(f \wedge g) = domain(f) \cup domain(g)$. We will frequently use the construction $(\hat{s} \otimes D)^{\top}$ and therefore introduce the more convenient notation

$(\hat{s} \otimes^{\top} D)$; note that this can also be represented by $(s \multimap D)$ the set of partial, but possibly empty, functions from s to D, with the ordering and $\wedge$ as defined above.

As mentioned in the introduction we will interpret the π-calculus in domains which have even more structure. Of course, to solve least fixpoints we require a least element in our domains and therefore will sometimes use the sub-category $l\mathscr{S}\mathscr{L}$ where the objects are domains; as usual we use $(\,)_{\perp}$ to denote the functor from $l\mathscr{S}\mathscr{L}_p$ which adjoins a bottom element to a predomains, with *down* representing its right adjoint. Note that in objects from $l\mathscr{S}\mathscr{L}$, $\wedge$ is always strict.

The meet operator $\wedge$ will be used to interpret *internal choice* but we will also require an interpretation for *external choice*, $+$, and the empty process 0. Unfortunately, we cannot work with distributive lattices as the operator $+$ will not behave, operationally, as a join operator; $p_1 + p_2$ is in general unrelated behaviourally to p_i. Nevertheless, we can salvage many of the properties of distributive lattices. A *lower choice predomain* is an object in $l\mathscr{S}\mathscr{L}_p$ endowed with an extra continuous binary operator $+$ and a constant 0 which satisfy the laws:

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

i.e. the extra structure is that of a commutative semi-group and both binary operators distribute over each other. We should point out that the last law given above is derivable from the others but nevertheless, because of the variation to be considered below, it is convenient to include it.

The initial object in the category of lower choice predomains (where the morphisms preserve all the operators), generated by a finite discrete set s is the set of *acceptance sets* over s, $\mathscr{A}(s)$. An acceptance set over s is a finite non-empty subset, $\mathscr{I}$, of the powerset of s which satisfies

- $J, K \in \mathscr{I}$ implies $J \cup K \in \mathscr{I}$,
- $J, K \in \mathscr{I}$, $J \subseteq L \subseteq K$ implies $L \in \mathscr{I}$.

Note that this definition is a slight generalisation of that used in [10] as we do not require $\cup\mathscr{I}$ to be s. In $\mathscr{A}(s)$ $+$ is defined pointwise, $\mathscr{I} + \mathscr{J} = \{J \cup K \mid J \in \mathscr{I},\ K \in \mathscr{J}\}$, while $\mathscr{I} \wedge \mathscr{J}$ is the smallest acceptance set containing both $\mathscr{I}$ and $\mathscr{J}$. However, we will tend to work in the more general category $l\mathscr{S}\mathscr{L}_p$ because some constructions will not preserve *external choice*, although all will preserve *internal choice*. Instead we will see that some constructions within $l\mathscr{S}\mathscr{L}_p$ will automatically lead to *choice predomains*. For example adjoining a top element to any object of $l\mathscr{S}\mathscr{L}_p$, constructs a choice predomain, since $+$ can be defined to be $\wedge$ and $\top$ can act as the 0.

These categories have variations, the categories $u\mathcal{S}\mathcal{L}_p$ and $u\mathcal{S}\mathcal{L}$, obtained by considering *upper* semi-lattices in place of lower semi-lattices; that is the objects are predomains, domains, respectively, endowed with a continuous binary operator $\vee$ which satisfies the laws of an upper-semilattice:

$$x \vee x = x$$
$$x \vee y = y \vee x$$
$$(x \vee y) \vee z = x \vee (y \vee z)$$
$$x \leqslant x \vee y$$

In objects from these categories all joins exist, since by definition all finite joins and all directed joins exist. An *upper* choice predomain is defined analogously; it is an object in $u\mathcal{S}\mathcal{L}_p$ with two extra continuous operators $+$ and 0 satisfying the laws given above. However, in this case much of the algebraic structure degenerates; it is straightforward to show that in any upper choice domain $\vee$ and $+$ coincide. Moreover, every upper choice predomain is automatically an upper choice domain as 0 is a minimal element.

We now turn our attention to the functor categories we require. Let I denote the category whose objects are finite subsets of the set of names $\mathcal{N}$ and whose morphisms are injections. This category has a symmetric monoidal structure; the tensor, which we denote by $+$ is given by disjoint union and the zero by the empty set. Moreover, it can be freely generated from one object 1 [19] which may be taken to be any any singleton set. Let $l\mathcal{D}_p$ denote the category whose objects are functors from I to $l\mathcal{S}\mathcal{L}_p$ 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 $l\mathcal{S}\mathcal{L}$, $u\mathcal{S}\mathcal{L}_p$, $u\mathcal{S}\mathcal{L}$, respectively, in place of $l\mathcal{S}\mathcal{L}_p$. These are autonomous categories as they inherit much of the structure of the underlying categories. Products, coproducts and $\otimes$ are defined pointwise but as usual exponentiation, $\Rightarrow$, the right-adjoint of $\otimes$, 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

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

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 $\mathbf{A}$ be the object in $l\mathcal{D}_p$ whose action on the finite set s gives the free (lower) choice predomain $\mathbf{A}(s+s)$, where $s+s$ is the disjoint union of s with itself, and which acts on the morphism $s \overset{i}{\mapsto} s'$ to give the obvious continuous function from $\mathbf{A}_s$ to $\mathbf{A}_{s'}$, defined by $\mathbf{A}(s \overset{i}{\mapsto} s')A = \{i(a) \mid a \in A\}$. The need for two copies of s will become apparent when we explain the construction of our model. The definition of the object $\mathbf{N}$, used to model names, is somewhat simpler: on the set s it gives the free object of $l\mathcal{S}\mathcal{L}_p$ 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 isomorphic to 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 continuous* functor [9] will have an initial fixpoint. In our domain equation we will use the locally continuous functors $\times$, $\otimes$ and $\Rightarrow$ together with the very simple one $\mathbf{s}$. It acts on a functor A as follows:

$$(\mathbf{s}A)_s = A_{(s+1)}$$
$$(\mathbf{s}A)_{(s \overset{i}{\mapsto} s')} = A_{(i+1)}$$

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 $s \overset{\text{in}}{\mapsto} s+1$ denotes the injection of s into $s+1$.

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

$$\mathbf{s}(A^{\top}) \simeq (\mathbf{s}(A))^{\top}$$
$$\mathbf{s}(A_{\perp}) \simeq (\mathbf{s}(A))_{\perp}$$
$$\mathbf{s}(A \times B) \simeq \mathbf{s}(A) \times \mathbf{s}(B)$$
$$\mathbf{s}(A + B) \simeq \mathbf{s}(A) + \mathbf{s}(B)$$
$$\mathbf{s}(\mathbf{N} \otimes A) \simeq (\mathbf{N} \otimes \mathbf{s}(A)) + \mathbf{s}(A)$$
$$\mathbf{s}(\mathbf{N} \Rightarrow A) \simeq \mathbf{s}(\mathbf{N}) \Rightarrow \mathbf{s}(A)$$

Notation. In general, we use $a : A \to B$ 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 $A \overset{a}{\mapsto} B$ and sometimes when working in a functor category we will use $a : A \dot{\to} B$ 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

Our language is a slight extension of the standard π-calculus [15, 16], with somewhat modified syntax. Let $\mathcal{V}$ be a set of process variables, ranged over by $X, Y, \ldots$, and $\mathcal{N}$

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

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

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

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

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

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

Fig. 1. Free variables in terms.

a countable set of names ranged over by x, y, n, m, ... Then the syntax of terms is given by

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

There are two binding operators for names, $\lambda x.-, \nu(n)-$, and one for process variables, $\mathsf{rec}\,\mathsf{X}.-$. Terms may have free occurrences of both and these are determined by the simple inference system given in Fig. 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. We assume standard notions of free and bound variables/names and α-equivalence. 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. A term p is *closed* if it contains no free occurrence of process variables, i.e. $\underline{x} \vdash p : P$. We will often refer to these as *process terms*, ranged over by $p, q, \ldots$. For such terms we let $fv(p)$ denote the set of names occurring freely in p, i.e. the names n such that $\underline{x} \vdash p : P$ implies n occurs in $\underline{x}$.

The operational semantics of the language is given as a reduction relation $\xrightarrow{\tau}$ between closed terms. The definition of $\xrightarrow{\tau}$ uses two auxiliary relations, $\xrightarrow{n?m}$, representing the ability to input the name m on channel n and $\xrightarrow{n!m}$ representing the ability to output m

Axioms:

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

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

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

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:

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

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

$$\frac{p \xrightarrow{\mu} p'}{v(x)p \xrightarrow{\mu} v(x)p'}\ x \text{ not in } \mu$$

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

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

Structural equivalence:

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

Fig. 2. Operational semantics.

along n. In the terminology of [15, 16] these are the free input and free output actions of processes. In Fig. 2 μ ranges over these three possible types of action $\tau, n?m$, and $n!m$; in later technical developments we will also use the so-called *bound* output actions.

As usual the operational semantics is expressed using a *structural congruence*, which is defined in the third part of Fig. 2. The operational semantics is then given as the least one satisfying the axioms and rules, also given in this figure. Note that

α-equivalence is included in the structural congruence and therefore operationally we work modulo α-equivalence. In the rule for the if then else statement we also assume 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 [17, 23] 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 *testing* 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 *success state* we introduce a *new* name ω, not occurring 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 must be e if for every maximal computation

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

there is some n such that $c_n \xrightarrow{\omega}$

Then $p \sqsubseteq_{\text{must}} q$ if for every test e, p must e implies q must 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 behaviour: $p \sqsubseteq_{\text{may}} q$ 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 $\sqsubseteq_{\text{must}}$ and in Section 6 we show how this can be modified so as to provide a semantics which is fully abstract with respect to $\sqsubseteq_{\text{may}}$.

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 $l\mathscr{D}_p$. To do so we use an object in the sub-category $l\mathscr{D}$, 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 $l\mathscr{D}_p$; here $\mathbf{N}^{\otimes \underline{x}}$ represents the product of copies of $\mathbf{N}$, one for each element in $\underline{x}$. Because of the nature of these products these morphisms have a simple representation in terms of *environments*. For any finite subset s of $\mathscr{N}$ let ENV_s be the set of mappings from $\mathscr{N}$ 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 $l\mathscr{S}\mathscr{L}_p$, 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 $\{\mathbf{P}_s \mid s \subseteq \mathscr{N}\}$ comes equipped with translation morphisms $(\mathbf{P}_\sigma) : \mathbf{P}_s \to \mathbf{P}_{s'}$, for each injection $s \overset{\sigma}{\mapsto} s'$ and these translation morphisms also relate the interpretation of terms in the different domains:

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

In order to give this interpretation we require, in addition to the object $\mathbf{P}$,

- $\oplus_{\mathbf{P}}$, a morphism from $\mathbf{P} \otimes \mathbf{P}$ to $\mathbf{P}$, to interpret internal choice. $\mathbf{P}$ will be a lower semi-lattice and $\oplus_{\mathbf{P}}$ be the meet operation $\wedge^{\otimes}$.
- $+_{\mathbf{P}}$, a morphism from $\mathbf{P} \otimes \mathbf{P}$ to $\mathbf{P}$, to interpret external choice. The lower semi-lattice $\mathbf{P}$ will be constructed in such a way that it is also a *choice domain* and will therefore come equipped with a natural candidate for $+_{\mathbf{P}}$, namely the extra binary operator $+^{\otimes}$.
- $0_{\mathbf{P}}$, an object in $\mathbf{P}$ to interpret the language construct 0. Again we will have a natural candidate since $\mathbf{P}$ will be a *choice domain*.
- $\|_{\mathbf{P}}$, a morphism from $\mathbf{P} \otimes \mathbf{P}$ to $\mathbf{P}$ to model the parallel construct.
- $\nu_{\mathbf{P}}$, a morphism from $(\mathbf{N} \Rightarrow \mathbf{P})$ to $\mathbf{P}$ to interpret the construct $\nu(n)t$.
- Abstractions will be interpreted in $(\mathbf{N} \Rightarrow \mathbf{P})$ and therefore to interpret the language construct $n?f$ we need a morphism $\mathbf{in}_{\mathbf{P}}$ from $\mathbf{N} \otimes (\mathbf{N} \Rightarrow \mathbf{P})$ to $\mathbf{P}$.
- In order to interpret the output construct $n!x.t$ we require a morphism $\mathbf{out}_{\mathbf{P}}$ from $\mathbf{N} \otimes \mathbf{N} \otimes \mathbf{P}$ to $\mathbf{P}$.

Because of the presence of recursive terms the definition of the semantics will also apply to open terms. So, in fact, we interpret the general type judgement $\Gamma \vdash t : P$ as a morphism from $[\![\Gamma]\!]$ to $\mathbf{P}$ where the object $[\![\Gamma]\!]$ is determined by the structure of Γ.

If it has the form $\underline{x}$; $\underline{X}$ then it has the form $\mathbf{N}^{\otimes^{\underline{x}}} \otimes \mathbf{P}^{\otimes^{\underline{X}}}$. Therefore, when t is a closed term this will be a morphism from $\mathbf{N}^{\otimes^{\underline{x}}}$ to $\mathbf{P}$, as promised above. The definition of $[\![\Gamma \vdash t : P]\!]$ is by induction on the inference of this judgement. For convenience we will use π_α to denote the projection from $[\![\Gamma]\!]$ unto the α component; α may be a name or a process variable:

- $[\![\underline{x}; X_1 \ldots X_i \ldots X_n \vdash X_i : P]\!] = \pi_{X_i}$.
- $[\![\ldots \beta\alpha \ldots \vdash t : P]\!] = [\![\ldots \alpha\beta \ldots \vdash t : P]\!] \circ \langle \pi_{x_1}, \ldots \pi_\beta, \pi_\alpha, \ldots, \pi_{X_n} \rangle$.
- $[\![\Gamma \vdash 0 : P]\!] = !_{[\![\Gamma]\!]} \circ 0_{\mathbf{P}}$.
- $[\![\Gamma + n \vdash n?f : P]\!] = \mathbf{in}_{\mathbf{P}} \circ \langle \pi_n, [\![\Gamma + n \vdash f : F]\!] \rangle$.
- $[\![\Gamma + n + x \vdash n!x.t : P]\!] = \mathbf{out}_{\mathbf{P}} \circ \langle \pi_n, \pi_x, [\![\Gamma + n + x \vdash t : P]\!] \rangle$.
- For $op = \|, +, \oplus$, $[\![\Gamma \vdash t \ op \ u : P]\!] = \mathbf{P}_{op} \circ \langle [\![\Gamma \vdash t : P]\!], [\![\Gamma \vdash u : P]\!] \rangle$.
- $[\![\Gamma \vdash \text{ if } b : t \text{ then } u \text{ else } P]\!] = \mathbf{if} \circ \langle [\![\Gamma \vdash b]\!], [\![\Gamma \vdash t : P]\!], [\![\Gamma \vdash u : P]\!] \rangle$.
  Here we assume the obvious semantic interpretation of booleans, $[\![\Gamma \vdash b]\!]$ in $([\![\Gamma]\!] \to \mathbf{Bool})$, where **Bool** is the functor whose action at every s gives the set of booleans $\{true, false\}$. Then **if** also acts pointwise; at every s it selects the second argument if the first is *true* and the third otherwise.
- $[\![\Gamma \vdash \lambda x.t : F]\!] = \lambda_{\mathbf{N}}([\![\Gamma + x \vdash t : P]\!])$.
  Note $[\![\Gamma + x \vdash t : P]\!]$ is a morphism in $([\![\Gamma]\!] \otimes \mathbf{N}) \to \mathbf{P}$ and since $\otimes$ is left adjoint to $\Rightarrow$ we have the mapping $\lambda_{\mathbf{N}}$ from $([\![\Gamma]\!] \otimes \mathbf{N}) \to \mathbf{P}$ to $[\![\Gamma]\!] \to (\mathbf{N} \Rightarrow \mathbf{P})$.
- $[\![\Gamma \vdash \nu(n)t : P]\!] = \nu_{\mathbf{P}} \circ \lambda_{\mathbf{N}}([\![\Gamma + n \vdash t : P]\!])$.
- $[\![\Gamma \vdash \mathsf{rec}\, \mathsf{X}.t : P]\!] = \mathbf{Y} \circ \lambda_{\mathbf{P}}([\![\Gamma + X \vdash t : P]\!])$.
  Here $[\![\Gamma + X \vdash t : P]\!]$ is a morphism in $([\![\Gamma]\!] \otimes \mathbf{P}) \to \mathbf{P}$ and using the adjunction between $\otimes$ and $\Rightarrow$ once more there is a mapping $\lambda_{\mathbf{P}}$ from $([\![\Gamma]\!] \otimes \mathbf{P}) \to \mathbf{P}$ to $[\![\Gamma]\!] \to (\mathbf{P} \Rightarrow \mathbf{P})$. The fixpoint operator used here, $\mathbf{Y}$, is a morphism from $(\mathbf{P} \Rightarrow \mathbf{P})$ to $\mathbf{P}$.

To complete our description of the interpretation we must exhibit the specific choice domain $\mathbf{P}$, an object in $l\mathscr{D}$, 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

$$\mathbf{P} \cong (\mathbf{A} \times (\mathbf{N} \otimes^{\top} \mathbf{F}) \times (\mathbf{N} \otimes^{\top} \mathbf{C}))_{\perp}$$

$$\mathbf{F} \cong \mathbf{N} \Rightarrow \mathbf{P}$$

$$\mathbf{C} \cong (\mathbf{N} \otimes \mathbf{P}) + \mathbf{s}(\mathbf{P})$$

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 $\mathscr{N}$. $\mathbf{P}_s$ is either the bottom element $\perp$ or has three components:

- $\mathbf{A}_s$, as acceptance set over $s + s$, representing the initial non-deterministic 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 $s + s$.
- $(\mathbf{N} \otimes^{\top} \mathbf{F})_s$, 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 $\mathscr{F}_s(\mathbf{P})$. 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.

- $(\mathbf{N} \otimes^{\top} \mathbf{C})_s$ 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:
  - $(\mathbf{N} \otimes \mathbf{P})_s$, 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 the subsequent behaviour of the process.
  - $\mathbf{s}(\mathbf{P})_s$, i.e. $\mathbf{P}_{(s+1)}$, representing the subsequent behaviour of the process after outputing an *arbitrary* new name not in s along the channel.

For convenience, whenever appropriate, we will elide the isomorphism pair *unfold* and *fold* between $\mathbf{P}$ and $(\mathbf{A} \times (\mathbf{N} \otimes^{\top} \mathbf{F}) \times (\mathbf{N} \otimes^{\top} \mathbf{C}))_{\perp}$. First note that since $\mathbf{P}$ is in the category $l\mathscr{D}$ it is a lower semi-lattice, thus providing an interpretation for $\oplus$. Moreover it is also a lower choice domain since

- $\mathbf{A}$ is a lower choice domain,
- both $\mathbf{N} \otimes^{\top} \mathbf{F}$ and $\mathbf{N} \otimes^{\top} \mathbf{C}$ are lower semi-lattices with zeros for $\wedge$, namely $\top$, and therefore are also lower choice domains,
- lower choice domains are closed under product,
- the lifting of a choice domain can be made into a choice domain by extending the operator $+$ in the obvious strict manner.

We therefore have the required interpretation for the language operators 0 and $+$.

To define the natural transformation $\mathbf{in}_{\mathbf{P}} : \mathbf{N} \otimes (\mathbf{N} \Rightarrow \mathbf{P}) \dot{\rightarrow} \mathbf{P}$ we must define for each s a morphism $(\mathbf{in}_{\mathbf{P}})_s$ from $(\mathbf{N} \otimes (\mathbf{N} \Rightarrow \mathbf{P}))_s$, i.e. $s \multimap^{\mathrm{ne}} (\mathbf{N} \Rightarrow \mathbf{P})_s$, to $\mathbf{P}_s$. This is defined as follows:

$$(\mathbf{in}_{\mathbf{P}})_s g = \wedge\{(\{\{n?\}\} \times (n \mapsto gn) \times \emptyset)_{\perp} \mid n \in domain(g)\}.$$

Intuitively, $(\mathbf{in}_{\mathbf{P}})_s$ is determined by its action on the functions with singleton domains. So for example when applied to the function $(n \mapsto f)$ it constructs the object $(\{n?\} \times (n \mapsto f) \times \emptyset)_{\perp}$.

The output natural transformation $\mathbf{out}_{\mathbf{P}}$ is defined in a similar manner. $(\mathbf{out}_{\mathbf{P}})_s$ is the morphism from $(\mathbf{N} \otimes \mathbf{N} \otimes \mathbf{P})_s$, i.e. $(s \times s \multimap^{\mathrm{ne}} \mathbf{P}_s)$, to $\mathbf{P}_s$ defined by

$$(\mathbf{out}_{\mathbf{P}})_s\, g = \wedge\{(\{\{n!\}\} \times \emptyset \times (n \mapsto (m \mapsto g(n, m))))_{\perp} \mid (n, m) \in domain(g)\}.$$

The definitions of the remaining two morphisms, $\|_{\mathbf{P}}$ and $\nu_{\mathbf{P}}$ are somewhat complicated. The definition of the first is not of great interest as it is determined by the requirement that the interleaving law in Fig. 5 be satisfied. It is therefore relegated to Appendix A. However the entire structure of $\mathbf{P}$, and its construction in the functor category, is motivated by the search for an adequate definition of $\nu_{\mathbf{P}}$. This is given in terms of an auxiliary natural transformation $\mathbf{new} : \mathbf{s}(\mathbf{P}) \dot{\rightarrow} \mathbf{P}$. Given $\mathbf{new}$ we can define the morphism $\nu_{\mathbf{P}}$, i.e. a natural transformation from $(\mathbf{N} \Rightarrow \mathbf{P})$ to $\mathbf{P}$, using a *coercion* natural transformation between $(\mathbf{N} \Rightarrow \mathbf{P})$ and $\mathbf{s}(\mathbf{P})$. Let $\mathbf{c} : (\mathbf{N} \Rightarrow \mathbf{P}) \dot{\rightarrow} \mathbf{s}(\mathbf{P})$ be given by the family of morphisms $\mathbf{c}_s$ from $(\mathbf{N} \Rightarrow \mathbf{P})_s$, i.e. $\mathscr{F}_s(\mathbf{P})$, to $(\mathbf{s}(\mathbf{P}))_s$, i.e. $\mathbf{P}_{(s+1)}$, defined

by $\mathbf{c}_s\, f = f(y)$ for any $y \notin s$. Note that this definition does not depend on the particular value of y chosen. Then the required natural transformation $v_{\mathbf{P}} : (\mathbf{N} \Rightarrow \mathbf{P}) \dot{\to} \mathbf{P}$ is given by $\mathbf{new} \circ \mathbf{c}$.

To define the natural transformation **new** from $\mathbf{s}(\mathbf{P})$ to $\mathbf{P}$ we make use of the properties of $\mathbf{s}$ given in Section 2. Using these one can easily show that $\mathbf{s}(\mathbf{P})$ is isomorphic to

$$(\mathbf{s}(\mathbf{A}) \times (\mathbf{s}(\mathbf{N} \otimes \mathbf{F}))^{\top} \times (\mathbf{s}(\mathbf{N} \otimes \mathbf{C}))^{\top})_{\perp}.$$

Therefore the definition of **new** will take the form $(\mathbf{new}_1 \times \mathbf{new}_2 \times \mathbf{new}_3)_{\perp}$ where

- $\mathbf{new}_1 : \mathbf{s}(\mathbf{A}) \to \mathbf{A}$,
- $\mathbf{new}_2 : (\mathbf{s}(\mathbf{N} \otimes \mathbf{F}))^{\top} \to \mathbf{N} \otimes^{\top} \mathbf{F}$,
- $\mathbf{new}_3 : \mathbf{s}(\mathbf{N} \otimes \mathbf{C}))^{\top} \to \mathbf{N} \otimes^{\top} \mathbf{C}$.

In fact, **new** will be used in the definition of $\mathbf{new}_i$ and therefore we will take **new** to be the *least* morphism to satisfy its definition. We define each of its components in turn:

- Intuitively, the action of $\mathbf{new}_1$ at the set s is to pointwise restrict subsets of $\mathscr{P}(s+1)$ to subsets of $\mathscr{P}(s)$. Formally, $(\mathbf{new}_1)_s$ is defined by $(\mathbf{new}_1)_s(\mathscr{I}) = \{J \backslash s \mid J \in \mathscr{I}\}$ where $J \backslash s = \{n?, n! \in J \mid n \in s\}$.
- To define $\mathbf{new}_2$ we use the isomorphism

  $$\mathbf{s}(\mathbf{N} \otimes \mathbf{F}) \simeq (\mathbf{N} \otimes \mathbf{s}(\mathbf{F})) + \mathbf{s}(\mathbf{F}).$$

  If $\mathbf{new}_f$ is a natural transformation from $\mathbf{s}(\mathbf{F})$ to $\mathbf{F}$ then $[id_{\mathbf{N}} \otimes \mathbf{new}_f, \lambda x.\top]$ can be taken to be a natural transformation from $\mathbf{s}(\mathbf{N} \otimes \mathbf{F})$ to $\mathbf{N} \otimes^{\top} \mathbf{F}$ and $\mathbf{new}_2$ will be defined 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 trivial extension of $[id_{\mathbf{N}} \otimes \mathbf{new}_c, \lambda x.\top]$ where $\mathbf{new}_c$ is a natural transformation from $\mathbf{s}(\mathbf{C})$ to $\mathbf{C}$. $\mathbf{s}(\mathbf{C})$ is isomorphic to $(\mathbf{N} \otimes \mathbf{s}(\mathbf{P})) + (\mathbf{s}(\mathbf{P}) + \mathbf{s}(\mathbf{s}(\mathbf{P})))$ and therefore $\mathbf{new}_3$ can be defined to be $(id_{\mathbf{N}} \otimes \mathbf{new}) + [id_{\mathbf{s}(\mathbf{P})}, \mathbf{s}(\mathbf{new}) \circ switch_{\mathbf{P}}]$, where $switch_{\mathbf{P}}$ is the natural transformation over $\mathbf{s}(\mathbf{s}(\mathbf{P}))$ which at s is given by the morphism $\mathbf{P}(id_s + switch)$, and $switch : 1 + 1 \to 1 + 1$ is the morphism in I which switches its arguments.

This completes our description of the interpretation of the π-calculus.

We have associated with each term p such that $\Gamma \vdash p : P$ a natural transformation between $[\![\Gamma]\!]$ and $\mathbf{P}$. As explained at the beginning of this section this gives, for each process term p, a family of interpretations $[\![p]\!]_s$ which can be viewed as mappings from ENV_s to $\mathbf{P}_s$. Specifically if $x_1 \ldots x_k \vdash p : P$ then for $\sigma \in ENV_s$ we let $[\![p]\!]_s\sigma$ denote the element in $\mathbf{P}_s$ given by $[\![p]\!]_s(\sigma(x_1) \otimes \cdots \otimes \sigma(x_k))$. With this notation the interpretation satisfies the well-known *Substitution Lemma*:

Lemma 4.1. $[\![p\sigma]\!]_s\rho = [\![p]\!]_s(\rho \circ \sigma)$.

5. Full abstraction

In this section we give the relationship between the operational and denotational semantics. Recall from the previous section that the denotational semantics gives rise to a family of interpretations,

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

one for each finite subset s of $\mathcal{N}$. Also each $\rho \in ENV_s$ can act as a substitution over processes. With this notation in mind the main result of this paper can be stated as:

Theorem 5.1. *For all process terms p, q and for every $\rho \in ENV_s$, $p\rho \sqsubseteq_{\text{must}} q\rho$ if and only if $[\![p]\!]_s\rho \leqslant [\![q]\!]_s\rho$.*

This immediately gives

Corollary 5.2. *For all process terms $p, q, [\![p]\!] \leqslant [\![q]\!]$ if and only if $p\rho \sqsubseteq_{\text{must}} q\rho$ for every environment ρ.*

We first give an outline of the structure of the proof of this result. This is followed by three subsections which contain the details. The essential ingredient of the proof is the notion of *acceptance traces*. Let $\mathtt{AccTr}$ be the least set such that

- $\Omega \in \mathtt{AccTr}$,
- $J \in \mathtt{AccTr}$ if J is a finite subset of $Ind = \{n?, n! \mid n \in \mathcal{N}\}$,
- $\mathtt{a} \in \mathtt{AccTr}$ implies:
  - $\mathtt{n?x.a} \in \mathtt{AccTr}$,
  - $\mathtt{n!x.a} \in \mathtt{AccTr}$,
  - $\nu(\mathtt{x})\mathtt{n!x.a} \in \mathtt{AccTr}$.

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 traces* and we will only consider them up to α-conversion. Note however that $n?x$ is NOT a binding operator; in the terminology of [15, 16] it represents a free input action.

Acceptance traces are ordered as follows:

- $\Omega \ll \mathtt{a}$ for every *acceptance trace* $\mathtt{a}$,
- $J \ll K$ if $J \subseteq K$,
- $\mathtt{a} \ll \mathtt{b}$ 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 $y \notin fv(\nu(\mathtt{x})\mathtt{b})$.

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

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

Sets of *acceptance traces* 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 $\xrightarrow{\tau}$ between process terms. Here we need to describe the *potential actions* a term can perform. There are three kinds of potential actions;

(1) an input action $\xrightarrow{n?x}$, as described in Section 3,

(2) an output action $\xrightarrow{n!x}$, also as described in Section 3,

(3) a *bound* output action $\xrightarrow{\nu(z)n!z}$.

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

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

These are extended to sequences over *Act* by

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

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

$$p \xrightarrow{\tau} p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_k \xrightarrow{\tau} \cdots$$

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

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

With this notation we can now define the behavioural acceptance traces associated with a process term:

- $p \models^{\mathrm{b}} \Omega$ if $p\Uparrow$,
- $p \models^{\mathrm{b}} J$ if $p\Downarrow$ and $J = S(q)$ for some q such that $p \stackrel{\varepsilon}{\Rightarrow} q$,
- $p \models^{\mathrm{b}} \alpha.\mathtt{a}$ if $p\Downarrow$ and $q \models^{\mathrm{b}} \alpha$ for some q such that $p \stackrel{\alpha}{\Rightarrow} q$.

Let $\mathtt{AccTr}^{\mathrm{b}}(p)$ denote the set of behavioural acceptance traces of p, $\{\mathtt{a} \mid p \models^{\mathrm{b}} \mathtt{a}\}$. These sets characterise the behavioural preorder $\sqsubseteq_{\mathrm{must}}$:

Theorem 5.3. *$p \sqsubseteq_{\mathrm{must}} q$ if and only if $\mathtt{AccTr}^{\mathrm{b}}(p) \ll \mathtt{AccTr}^{\mathrm{b}}(q)$.*

The proof of this theorem is given in Section 5.3.

In a similar manner we can associate with each element d of the domain $\mathbf{P}_s$ a set of acceptance traces $\mathtt{AccTr}_{\mathrm{sem}}(d)$; the details are postponed until Section 5.1 because the structure of $\mathbf{P}_s$ needs to be elaborated.

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

Theorem 5.4. *For every $d, e \in \mathbf{P}_s$, $d \leqslant e$ if and only if $\mathtt{AccTr}_{\mathrm{sem}}(d) \ll \mathtt{AccTr}_{\mathrm{sem}}(e)$.*

The proof of this theorem will be given in the Section 5.1.

The third major result relates the behavioural *acceptance traces* of a process term with the *acceptance traces* of its denotation. In general these will not be identical.

For example it turns out that the only sets of indications in $\mathtt{AccTr}^{\mathrm{b}}(n!x.0 + m!y.0)$ are $\{n!\}$ and $\{m!\}$ but, from the definition of $\mathtt{AccTr}_{\mathrm{sem}}$ to be given in the next subsection, $\{n!, m!\}$ is also in $\mathtt{AccTr}_{\mathrm{sem}}([\![n!x.0 + m!y.0]\!]_s)id$, where $s = \{n, m\}$. However, the sets of behavioural *acceptance traces* and denotational *acceptance traces* 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 5.5. *Suppose* $x_1 \ldots x_k \vdash p : P$. *Then for every* s *such that* $x_i \in s$, $\mathtt{AccTr}^{\mathrm{b}}(p) \cong \mathtt{AccTr}_{\mathrm{sem}}([\![p]\!]_s id)$ *where* $id \in ENV_s$ *is any environment which is the identity on* s.

Section 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

$$\begin{aligned} p\rho \sqsubseteq_{\mathrm{must}} q\rho &\Leftrightarrow AccTr^{\mathrm{b}}(p\rho) \ll AccTr^{\mathrm{b}}(q\rho) \quad \text{by Theorem 5.3} \\ &\Leftrightarrow AccTr_{\mathrm{sem}}([\![p\rho]\!]_s id) \ll AccTr_{\mathrm{sem}}([\![q\rho]\!]_s id) \text{ by Theorem 5.5} \\ &\Leftrightarrow [\![p\rho]\!]_s id \leqslant [\![q\rho]\!]_s id \text{ by Theorem 5.4} \\ &\Leftrightarrow [\![p]\!]_s \rho \leqslant [\![q]\!]_s \rho \end{aligned}$$

The last line is an application of the *Substitution 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 traces* 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 $(\mathbf{N} \otimes^{\top} \mathbf{H})_s$ can be represented as $(s \rightharpoonup \mathbf{H}_s)$. Moreover if $\sigma : \mathbf{H} \dot{\rightarrow} \mathbf{K}$ then the action of the natural transformation $(id \otimes^{\top} \sigma) : \mathbf{N} \otimes^{\top} \mathbf{H} \dot{\rightarrow} \mathbf{N} \otimes^{\top} \mathbf{K}$ at s is given, in terms of this representation, by the morphism $(\mathbf{N} \otimes^{\top} \mathbf{H})_s : (s \rightharpoonup \mathbf{H}) \rightarrow (s \rightharpoonup \mathbf{K})$, 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 $\mathscr{F}_s(\mathbf{H})$ with domain $\mathscr{N}$. We can also describe the actions of natural transformations in terms of these representations. If $\sigma : \mathbf{H} \dot{\rightarrow} \mathbf{K}$ then the action of the natural transformation $(id \Rightarrow \sigma) : (\mathbf{N} \Rightarrow \mathbf{H}) \dot{\rightarrow} (\mathbf{N} \Rightarrow \mathbf{K})$ at s is given by the morphism $(id \Rightarrow \sigma)_s : \mathscr{F}_s(\mathbf{H}) \rightarrow \mathscr{F}_s(\mathbf{K})$, 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 $(s \rightharpoonup^{\mathrm{ne}} \mathbf{H}_s)$ and $\mathbf{s}(\mathbf{H})_s$ is determined by an 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 $\mathscr{N}$. Let $\mathscr{P}\mathscr{F}_s(\mathbf{H})$ 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: $f \leqslant g$ if
- $domain(g) \leqslant domain(f)$,
- for all $n \in domain(g)$, $f(n) \leqslant g(n)$.

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 $\sigma : \mathbf{H} \dot{\to} \mathbf{K}$. Then the action of the natural transformation

$$(id \otimes \sigma + \mathbf{s}(\sigma)) : (\mathbf{N}\otimes\mathbf{H} + \mathbf{s}(\mathbf{H})) \dot{\to} (\mathbf{N}\otimes\mathbf{K} + \mathbf{s}(\mathbf{K}))$$

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 $\mathcal{I}_s(d)$ denote $(\pi_1 \circ down)d$; $\mathcal{I}_s(d)$ 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 traces* with elements of $\mathbf{P}_s$. For any $d \in \mathbf{P}_s$ let

(1) $d \models_s \Omega$ if $unfold(d) = \perp$,

(2) if $unfold(d) \neq \perp$:
- $d \models_s J$ if $J \in \mathtt{AccTr}_{\mathrm{sem}}(d)$,
- $d \models_s \mathtt{n?x.a}$ if $d^? n\, x \models_{s\cup\{x\}} \mathtt{a}$,
- for $x \in s$, $d \models_s \mathtt{n!x.a}$ if $d^! n\, x \models_s \mathtt{a}$,
- $d \models_s \nu(\mathtt{y})\mathtt{n!y.a}$ 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 acceptance traces can be transferred from one to the other. Any $s \overset{\sigma}{\mapsto} s'$ can also be viewed as a substitution over names; it leaves untouched all those which do not appear in its domain s. So, viewing acceptance traces as simple terms, this means that these injections can also be applied to acceptance traces. With this notation we have

Proposition 5.6. *If* $s \overset{\sigma}{\mapsto} s'$ *then* $d \models_s \mathtt{a}$ *implies* $\mathbf{P}_\sigma(d) \models^{s'} \mathtt{a}\sigma$.

Proof. By induction on the structure of 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 acceptance traces are given in the following lemma:

Lemma 5.7. (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 $\mathscr{P}\mathscr{F}_s(\mathbf{P})$; in particular the fact that $f(n) = \mathbf{P}_{(s+m\mapsto n)}f(m)$. □

Let $\mathtt{AccTr}_{\mathrm{sem}}(d)$ denote the set of semantic acceptance traces of the element d, $\{\mathtt{a} \mid d \models_s \mathtt{a}\}$.

Proposition 5.8. *For all* $d, e \in \mathbf{P}_s$, $d \leqslant e$ *implies* $\mathtt{AccTr}_{\mathrm{sem}}(d) \ll \mathtt{AccTr}_{\mathrm{sem}}(e)$.

Proof. It is straightforward to show by induction on $\mathtt{a}$ that $\forall s\ \forall d, e \in \mathbf{P}_s$, $d \leqslant e$, $e \models_s \mathtt{a}$ 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 its 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:

$$\mathbf{pid}^k : \mathbf{P} \dot{\to} \mathbf{P}$$
$$\mathbf{fid}^k : \mathbf{F} \dot{\to} \mathbf{F}$$
$$\mathbf{cid}^k : \mathbf{C} \dot{\to} \mathbf{C}$$

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) $\mathbf{fid}^{k+1} = (id \Rightarrow \mathbf{pid}^k)$,
(3) $\mathbf{cid}^{k+1} = (id \otimes \mathbf{pid}^k) + \mathbf{s}(\mathbf{pid}^k)$,
(4) $\mathbf{pid}^{k+1} = (id \times (id \otimes^{\top} \mathbf{fid}^{k+1}) \times (id \otimes^{\top} \mathbf{cid}^{k+1}))_{\bot}$.
We state without proof:

Theorem 5.9. *For* $\mathbf{A} = \mathbf{P}, \mathbf{C}, \mathbf{F}$, $\vee_k(\mathbf{aid}^k) = id_{\mathbf{A}}$.

With this characterisation of the domains we can now prove

Proposition 5.10. *For all* $d, e \in \mathbf{P}_s$, $\mathtt{AccTr}_{\mathrm{sem}}(d) \ll \mathtt{AccTr}_{\mathrm{sem}}(e)$ *implies* $d \leqslant e$.

Proof. From the previous theorem it is sufficient to show that for all k, $\mathbf{pid}^k_s(d) \leqslant \mathbf{pid}^k_s(e)$. 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 $\mathtt{AccTr}(e) \ll \mathtt{AccTr}(d)$ this also means that $e \neq \bot$. Using the representations developed above this means that $down(\mathbf{pid}_s^{k+1}(d))$ and $down(\mathbf{pid}_s^{k+1}(e))$ may be taken to be

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

respectively. Now $\mathtt{AccTr}_{\mathrm{sem}}(e) \ll \mathtt{AccTr}_{\mathrm{sem}}(d)$ means that $\cup\mathscr{I}_s(e) \subseteq \cup\mathscr{I}_s(d)$ and therefore $\mathscr{I}_s(e) \subseteq \mathscr{I}_s(d)$, since the latter are acceptance sets. It remains to prove

$$\mathbf{fid}_s^{k+1} \circ d^? \leqslant \mathbf{fid}_s^{k+1} \circ e^?$$
$$\mathbf{cid}_s^{k+1} \circ d^! \leqslant \mathbf{cid}_s^{k+1} \circ e^!$$

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 $\mathtt{AccTr}_{\mathrm{sem}}(e) \ll \mathtt{AccTr}_{\mathrm{sem}}(d)$ 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

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

as objects in $\mathbf{C}_s$, i.e $\mathscr{P}\mathscr{F}_s(\mathbf{P})$. 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

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

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 $e \models_s \mathtt{n!m.a}$ for some $\mathtt{a}$. For any $m \notin s$, $m \in domain(e^!n)$ if and only if $e \models_s \nu(\mathtt{z})\mathtt{n!z.a}$ for some $\mathtt{a}$. Therefore, the required property follows from $\mathtt{AccTr}(e) \ll \mathtt{AccTr}(d)$.
- for every $m \in domain(e^!n)$, $\mathbf{pid}^k(d^!nm) \leqslant \mathbf{pid}^k(d^!nm)$.

  First suppose $m \in s$. Then using part 2 of Lemma 5.7 we have that $\mathtt{AccTr}_{\mathrm{sem}}(e) \ll \mathtt{AccTr}_{\mathrm{sem}}(d)$ implies $\mathtt{AccTr}_{\mathrm{sem}}(e^!nm) \ll \mathtt{AccTr}_{\mathrm{sem}}(d!nm)$. So we can apply induction to obtain the required result.

  If $m \notin s$ then, using the third part of the same lemma, we have $\mathtt{AccTr}_{\mathrm{sem}}(e^!nm) \ll \mathtt{AccTr}_{\mathrm{sem}}(d^!nm)$, since $e \models_s \nu(\mathtt{z})\mathtt{n!z.a}$ if and only if $e^!nm \models_{s\cup\{m\}} \mathtt{a}[m/z]$, and once more it follows by induction that $\mathbf{pid}^k(d^!nm) \leqslant \mathbf{pid}^k(d^!nm)$. □

Combining these two propositions we get Theorem 5.4:

Theorem 5.11 (Internal full abstraction). *For every* $d, e \in \mathbf{P}_s$, $d \leqslant e$ *if and only if* $\mathtt{AccTr}_{\mathrm{sem}}(d) \ll \mathtt{AccTr}_{\mathrm{sem}}(e)$.

We should point out that the sets of acceptance traces generated behaviourally by processes are more restricted than those generated semantically by elements in the models $\mathbf{P}_s$; they are in some intuitive sense more coherent. For example if, for some process

p, $n!m \in J$ for some $J \in \mathtt{AccTr}^{\mathrm{b}}(p)$ then we also have that $\mathtt{n!m.a}$ is in $\mathtt{AccTr}^{\mathrm{b}}(p)$ for some acceptance trace $\mathtt{a}$. This is not in general true of $\mathtt{AccTr}_{\mathrm{sem}}(d)$ for $d \in \mathbf{P}_s$ as we have not placed any coherency constriants on the three components of the domain $\mathbf{P}$. This means that there are primes in $\mathbf{P}_s$ which are not denotable by terms in our language but luckily their presence do not impinge on full abstractness.

5.2. Relating behavioural and denotational acceptance traces

The central technical device in this section is that of *head normal form* (*hnf*). We show that, using equations which are sound for both the behavioural preorder and the order induced by the denotational model on process terms, every convergent term can be reduced to a *hnf*. The structure of these *hnf*'*s* are such that it is relatively straightforward to relate their behavioural and denotational *acceptance traces*.

The set of basic inequations are given in Fig. 3, where for convenience we add a new constant to the language, Ω, to denote $\mathtt{rec\,X.}\ X$. These include the defining equations for *choice domains* together with many standard equations associated with the π-calculus. A typical example of the latter is

$$v(n)m?\lambda y.X = m?\lambda y.v(n)X \quad \text{if } n \neq y,\ n \neq m$$

Others are more commonly associated with testing style equivalences and typical examples of these include:

$$n?\lambda x.X + n?\lambda x.Y = n?\lambda x.X \oplus n?\lambda x.Y$$
$$n!x.X + n!x'.Y = n!x.X \oplus n!x'.Y$$

Some care must be taken when applying these to process terms. In particular, they can not be applied under input prefixes as the inference rule

$$\frac{p \leqslant q}{n?\lambda x.p \leqslant n?\lambda x.q}$$

is not in general sound for $\sqsubseteq_{\mathrm{must}}$. For example, let p, q denote

$$m?\lambda z.x!w.0 + x!w.m?\lambda z.0, \quad (m?\lambda z.0) \,||\, (x!w.0)$$

respectively. Then $p \sqsubseteq_{\mathrm{must}} q$ but $n?\lambda x.p \not\sqsubseteq_{\mathrm{must}} n?\lambda x.q$ because $n?\lambda x.q$ can perform the action $n?m$ followed by a communication to be transformed into 0 whereas $n?\lambda x.p$ does not have this possibility. So $n?\lambda x.p$ must satisfy the test $n!m.m?\lambda x.\omega$ whereas $n?\lambda x.q$ may fail it. This rule is sound for the denotational semantics, i.e. $[\![p]\!] \leqslant [\![q]\!]$ implies $[\![n?\lambda x.p]\!] \leqslant [\![n?\lambda x.q]\!]$. However, we are trying to relate the behaviour of a process term p, as captured by $\mathtt{AccTr}^{\mathrm{b}}(p)$, with a particular component of its denotation $[\![p]\!]$, namely $[\![p]\!]_s id$, and the rule becomes unsound when applied to this component of the denotation.

The proof system we use for manipulating terms is given in Fig. 5 and the only difference from the standard equational proof system is the more complicated input rule; to infer $n?\lambda x.p \leqslant n?\lambda x.q$ we must establish $p[z/x] \leqslant q[z/x]$ for *every* name z (although

$$
\begin{aligned}
X \oplus X &= X \\
X \oplus Y &= Y + X \\
(X \oplus Y) \oplus Z &= X \oplus (Y \oplus Z) \\
X \oplus Y &\leqslant X \\
X + X &= X \\
X + Y &= Y + X \\
(X + Y) + Z &= X + (Y + Z) \\
X + 0 &= X \\
X + (Y \oplus Z) &= (X + Y) \oplus (X + Z) \\
X \oplus (Y + Z) &= (X \oplus Y) + (X \oplus Z) \\
X \oplus Y &\leqslant X + Y \\
n?\lambda x.X + n?\lambda x.Y &= n?\lambda x.(X \oplus Y) \\
n!x.X + n!x.Y &= n!x.(X \oplus Y) \\
n?\lambda x.X + n?\lambda x.Y &= n?\lambda x.X \oplus n?\lambda x.Y \\
n!x.X + n!x'.Y &= n!x.X \oplus n!x'.Y \\
\nu(n)(X + Y) &= \nu(n)X + \nu(n)Y \\
\nu(n)(X \oplus Y) &= \nu(n)X \oplus \nu(n)Y \\
\nu(n)n?X &= 0 \\
\nu(n)n!x.X &= 0 \\
\nu(n)m?\lambda y.X &= m?\lambda y.\nu(n)X \quad \text{if } n \neq y,\ n \neq m \\
\nu(n)m!y.X &= m!y.\nu(n)X \quad \text{if } n \neq y,\ n \neq m \\
\nu(n)\nu(m)X &= \nu(m)\nu(n)X \\
(X \oplus Y) \parallel Z &= (X \parallel Z) \oplus (Y \parallel Z) \\
X \parallel Y &= Y \parallel X \\
X + \Omega &\leqslant X \oplus \Omega \\
X \parallel (Y + \Omega) &= \Omega \\
\Omega &\leqslant X \\
\nu(n)\Omega &= \Omega \\
\mathsf{rec}\,\mathsf{X}.t &= t[\mathsf{rec}\,\mathsf{X}.t/X]
\end{aligned}
$$

Fig. 3. Equations.

it will well known [21] that it is possible to restrict this to a finite number). We write $\vdash p \leqslant q$ if we can derive $p \leqslant q$ in the proof system using the basic equations and the interleaving law in Fig. 4. It is straightforward to establish

Proposition 5.12. *$\vdash p \leqslant q$ implies*
(1) $p \ll q$,
(2) $[\![p]\!]_s id \leqslant [\![q]\!]_s id$ *for any* $id \in ENV_s$ *which is the identity on* s.

Note that in general $\vdash p \leqslant q$ does not mean $[\![p]\!]_s \sigma \leqslant [\![q]\!]_s \sigma$ for every σ. In particular the interleaving law is not sound when interpreted in this manner. For example let p, q

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

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

where

$$ext(p, q) = \sum\{pre_i \cdot (p_i \parallel q,\ i \in I\} + \sum\{pre_j \cdot (p \parallel q_j),\ j \in J\}$$

$$int(p, q) = \circ\sum\{r_{i,j},\ comm(i, j)\}$$

where $comms(p, q)$ is true if there is some i, j such that $comm(i, j)$ and $comm(i, j)$ holds when

(1) pre_i is $n!x$ and pre_j is $n?\lambda y.$; then $r_{i,j}$ is $p_i \parallel (q_j[x/y])$
(2) pre_i is $v(x)$ $n!x$ and $pre_j$is $n?\lambda y.$; then $r_{i,j}$ is (w) $(p_i[w/x] \parallel q_j[w/y])$ where w is not free in $v(x)$ p_i or in $v(y)q_j$
(3) pre_i is $n?\lambda y.$ and pre_j is $n!x$: then $r_{i,j}$ is $p_i[x/y] \parallel q_j$
(4) pre_i is $n?\lambda y.$ and pre_j is $v(x)$ $n!$; then $r_{i,j}$ is $(w)(p_i[w/y]) \parallel q_j[w/x])$ where w is not free in $v(y)p_i$ or in $v(x)$ q_j

Fig. 4. Interleaving law.

be the processes defined above,

$$m?\lambda z.x!w.0 + x!w.m?\lambda z.0, \quad (m?\lambda z.0) \parallel (x!w.0)$$

and let $\sigma(m) = \sigma(n) = k$. Then $\vdash p \leqslant q$ but $[\![p]\!]_s\sigma \not\leqslant [\![q]\!]_s\sigma$.

We now explain the syntax of *hnfs*.

Definition 2. Let $\mathscr{I}$ be an acceptance set over Ind. For each $c \in \cup\mathscr{I}$ let p_c be a process term given as follows:

- If c is $n?$ then p_c has the form $n?p^{n?}$ where $p^{n?}$ is some abstraction term.
- c has the form $n!$ then p_c has the form

$$n!y_1.p^{(n!y_1)} + \cdots + n!y_k.p^{(n!y_k)} \quad \{+v(y)n!y.p^{(v,y)}\}$$

where $p^{(n!y_i)}$, $p^{(v,y)}$ are arbitrary process terms, y_i, y are distinct names and $\{+\cdots\}$ indicates an optional term.

Then

$$\circ \sum_{J \in \mathscr{I}} \left\{ \sum_{c \in J} \{p_c\} \right\}$$

is a *hnf*.

Proposition 5.13. *If $p \Downarrow$ then there exists a hnf, $hnf(p)$ such that $\vdash p = hnf(p)$.*

I $\dfrac{}{p \leqslant p}$ $\qquad$ $\dfrac{p \leqslant q,\ q \leqslant r}{p \leqslant r}$

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

$op = +, \oplus \parallel v(n)XS$

Eq $\dfrac{}{p \leqslant q}$

for every instance of an inequation

α $\dfrac{p \equiv_\alpha q}{p = q}$

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

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

$\dfrac{p' \leqslant q'}{\text{if } b \text{ then } p \text{ else } p' \leqslant \text{if } b \text{ then } q \text{ else } q'}$ $\qquad$ $\dfrac{[\![b]\!] = false}{\text{if } b \text{ then } X \text{ else } Y = Y}$

Fig. 5. The proof system.

Proof. It is virtually identical to that of Proposition 4.3 of [12] and is therefore omitted. The new operator $v(n)$ is accommodated by the third group of equations in Fig. 3; if h is a *hnf* then these can be used to reduce $v(n)\ h$ to a *hnf*. The only new ingredient in the proof is to ensure that in the subterms $h_{n!}$ there is at most one summand of the form $v(x)\ n!x.q$. If during the reduction procedure two such summands are generated then they can be replaced by one using the equations as follows:

$$\begin{aligned} v(x)\ n!x.q + v(y)\ n!y.q' &= v(z)\ n!z.(q[z/x]) + v(z)\ n!z.(q'[z/y]) \\ &\quad \text{by } \alpha\text{-conversion, where } z \text{ is new} \\ &= v(z)\ (n!z.q[z/x] + n!z.q'[z/y]) \\ &= v(z)\ n!z.(q[z/x] \oplus q'[z/y]) \end{aligned}$$

Hnfs play a crucial role in

Lemma 5.14. *If $\underline{x} \vdash p : P$, s contains all x_i and id is any environment in ENV_s which is the identity on s then $p \Downarrow$ if and only if $[\![p]\!]_s id \neq \perp$.*

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

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

Now suppose $[\![p]\!]_s id \neq \perp$. Let p^n, $n \geqslant 0$, be the infinite sequence of *recursion free* terms obtained by unrolling occurrences of sub-terms of the form $\mathsf{rec}\,X.t$. Then standard techniques from denotational semantics can be used to show that $[\![p]\!] = \vee_n [\![p^n]\!]$ which in turn means that for some k $[\![p^k]\!]_s id$ is different from $\perp$. From the above remark we therefore have that $p^k \Downarrow$. Now $p^k \leqslant p$ can be derived in the proof system and since this is sound with respect to $\ll$ it follows that $p \Downarrow$. □

The syntactic structure of *hnf*'s 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 $\underline{x}$; we also use $\mathscr{I}(h)$ to denote the acceptance set $\mathscr{I}$ used in the definition. 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 5.15. $\mathtt{AccTr}_{\mathrm{sem}}([\![h]\!]_s \sigma) = \sigma(\mathscr{I}(h))$.

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

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 $m? \in \cup \mathscr{I}(h)$, $h^{m?}$ is an abstraction term and therefore $[\![h^{m?}]\!]_s \sigma \in \mathbf{F}_s$. If $\sigma(m) = n$ then

Lemma 5.16. $([\![h]\!]_s \sigma)^? n = [\![h^{m?}]\!]_s \sigma$.

Proof. Again a simple examination of the interpretation of *hnf*'s. □

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 $\mathscr{PF}_s(\mathbf{P}_s)$, can be defined using $h^{m!}$. Suppose this has the form

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

Let $dec_{m!}(h) \in \mathscr{PF}(\mathbf{P}_s)$ be defined by

$$dec_{m!}(h) z = \begin{cases} [\![h_i]\!]_s \sigma, & z = \sigma(y_i) \\ [\![h_y]\!]_{s \cup \{z\}} \sigma[z/y], & z \notin s \end{cases}$$

Again an examination of the interpretation of *hnf*'s shows that

Lemma 5.17. $([\![h]\!]_s \sigma)^! n = dec_{m!}(h)$.

These lemmas make straightforward the proof of the main theorem of this section:

Theorem 5.18 (Theorem 5.5). *Suppose* $x_1 \ldots x_k \vdash p : P$. *Then for every* s *such that* $x_i \in s$, $\mathtt{AccTr}^{\mathtt{b}}(p) \cong \mathtt{AccTr}_{\mathrm{sem}}([\![p]\!]_s id)$ *where* $id \in ENV_s$ *is any environment which is the identity on* s.

Proof. We prove by structural induction on $\mathtt{a}$ that

(1) $p \models^{\mathtt{b}} \mathtt{a}$ implies $[\![p]\!]_s id \models_s \mathtt{a}'$ for some $\mathtt{a}'$ such that $\mathtt{a}' \ll \mathtt{a}$.

(2) $[\![p]\!]_s id \models_s \mathtt{a}$ implies $p \models^{\mathtt{b}} \mathtt{a}'$ for some $\mathtt{a}'$ such that $\mathtt{a}' \ll \mathtt{a}$.

If $\mathtt{a} = \Omega, J$ then this follows from Lemmas 5.14 and 5.15, respectively. We consider two other cases, when $\mathtt{a}$ has the form $\mathtt{n?y.b}$, $v(\mathtt{m})\ \mathtt{n!m.b}$, respectively, and in both we may assume that p is a *hnf*':

(1) Suppose $p \models^{\mathtt{b}} \mathtt{n?y.b}$. Let $p^{n?}$ have the form $\lambda x.t$. This means that $t[y/x] \models^{\mathtt{b}} \mathtt{b}$. 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 *Substitution 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 $\mathscr{F}(\mathbf{P}_s)$, using Lemma 5.16, we obtain the function

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

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 $[\![p]\!]_s id \models_s v(\mathtt{m})\mathtt{n!m.b}$ and for convenience let us assume that $m \in s$. Using Lemma 5.17 this means that for all $z \notin s$

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

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 *Substitution 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 $p_y[m/y] \models^{\mathtt{b}} \mathtt{b}'$ and since $p \xrightarrow{v(m)n!m} p_y[m/y]$ it follows that $p \models v(\mathtt{m})\mathtt{n!m.}\ \mathtt{b}'$.

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 $\vdash d \leqslant q$ if and only $[\![d]\!]_s id \leqslant [\![q]\!]_s id$, 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 traces* 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 its properties. However, the following can be proved by proof induction on the definition of $\xrightarrow{\tau}$:

Lemma 5.19. *$p \xrightarrow{\tau} q$ implies $p\sigma \xrightarrow{\tau} q\sigma$ for every substitution σ.*

The following lemma will assist with the behavioural analysis.

Lemma 5.20. *$p\|q \xrightarrow{\tau} r$ if and only if one of the following*:
(1) $r \equiv p'\|q$ *where* $p \xrightarrow{\tau} p'$,
(2) $r \equiv p\|q'$ *where* $q \xrightarrow{\tau} q'$,
(3) $r \equiv p'\|q'$ *where* $p \xrightarrow{n?y} p'$ *and* $q \xrightarrow{n!y} q'$, *or* $q \xrightarrow{n?y} q'$ *and* $p \xrightarrow{n!y} p'$,
(4) $r \equiv (\nu(z)p'\|q')$ *for some* $z \notin fv(p,q)$ *and*

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

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

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

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) $\mathtt{a} = \Omega$:

$$t(\mathtt{a})^J_X = 1.\omega \text{ (an abbreviation for } \omega \oplus \omega)$$

(2) $\mathtt{a} = K$:

$$t(\mathtt{a})^J_X = \sum\{n!y.\omega \mid n! \in J\} + \sum\{n?\lambda y.\omega \mid n? \in J\}$$

(3) $\mathtt{a} = \mathtt{n?y.b}$:

$$t(\mathtt{a})^J_X = 1.\omega + n!y.t(\mathtt{b}^J_{X\cup\{y\}})$$

(4) $\mathtt{a} = \mathtt{n!y.b}$:

$$t(\mathtt{a})^J_X = 1.\omega + n?\lambda z.\ \mathsf{if}\ z = y\ \mathsf{then}\ t(\mathtt{b}^J_X)\ \mathsf{else}\ \omega, \text{ where } z \notin fv(t(\mathtt{b}^J_{X\cup\{y\}}, y))$$

(5) $\mathtt{a} = \nu(y)\mathtt{n!y.b}$:

$$t(\mathtt{a})^J_X = 1.\omega + n?\lambda z.\ \mathsf{if}\ z \in X\ \mathsf{then}\ w\ \mathsf{else}\ t(\mathtt{b}^J_{X\cup\{y\}}),$$
$$\text{where } z \notin fv(t(\mathtt{b})^J_{X\cup\{y\}}, y))$$

and $z \in X$ is an abbreviation for $z = x_1 \vee \cdots \vee x_k$.

We leave the reader to show that these tests satisfy the following properties:

Lemma 5.21. *For every* $\mathtt{a}$
(1) p *must* $t(\mathtt{a})^J_X$ *implies that* p *must* $t(\mathtt{a})^{J'}_X$ *for every* $J' \supseteq J$,
(2) *for every injective substitution* σ, $(t(\mathtt{a})^J_X)\sigma = t(\mathtt{a}\sigma)^{J\sigma}_{X\sigma}$.

The choice of the parameters to these tests, X and J, will depend on the free names occurring in the process term under consideration, and the set of *indications* associated with the *acceptance trace* being examined. The latter is defined by

- $\mathsf{Ind}(\mathtt{Omega}) = \emptyset$,
- $\mathsf{Ind}(K) = K$,
- $\mathsf{Ind}(\alpha.\mathtt{b}) = \mathsf{Ind}(\mathtt{b})$.

Proposition 5.22. *If* $fv(p) \subseteq X$ *then* ($\forall \mathtt{a}'$ *such that* $\mathtt{a}' \ll \mathtt{a}$, $p \not\models^{\mathtt{b}} \mathtt{a}'$) *if and only if there exists some* J *such that* $J \cap \mathsf{Ind}(\mathtt{a}) = \emptyset$ *and* p *must* $t(\mathtt{a}^J_X)$.

Proof. For convenience let us write $p\surd$ to indicate that for every computation

$$p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} p_k \xrightarrow{\tau} \ldots$$

there is some k such that $p_k \xrightarrow{\omega}$. Then p *must* e if and only $(p \parallel e)\surd$. We also let $Ls(\mathtt{a})$ denote the set $\{\mathtt{a}' \mid \mathtt{a} \ll \mathtt{a}'\}$.

The proof is by induction on $\mathtt{a}$. We examine four cases:

- $\mathtt{a} = \Omega$: $p \models^{\mathtt{b}} \mathtt{a}'$ for no $\mathtt{a}' \in Ls(\mathtt{a})$ if and only if $p \Downarrow$ if and only if $(p \parallel 1.\omega)\surd$ if and only if p *must* $t(a)^J_X$ for all J.
- $\mathtt{a} = K$: First suppose that $p \models^{\mathtt{b}} \mathtt{a}'$ for no $\mathtt{a}' \in Ls(\mathtt{a})$. This means $p \not\models^{\mathtt{b}} \Omega$, i.e. $p \Downarrow$, and also that whenever $p \xrightarrow{\tau}{}^* q$ then there is some indication i_q in $S(q) - K$. The required J is $\{i_q \mid p \xrightarrow{\tau}{}^* q\}$.

  Conversely suppose $p \parallel t(\mathtt{a})^J_X$ for some J such that $J \cap A(\mathtt{a}) = \emptyset$. This means that
  - $p \Downarrow$, i.e. $p \not\models^{\mathtt{b}} \Omega$
  - for any q such that $p \xrightarrow{\tau}{}^* q$, $S(q) \cap J \neq \emptyset$; this in turn means that $p \models \mathtt{K}'$ for no $K' \subseteq J$

  Combining these we obtain that $p \models^{\mathtt{b}} \mathtt{a}'$ for no $\mathtt{a}' \in Ls(\mathtt{a})$.
- $\mathtt{a} = \mathtt{n?y.b}$: First suppose that $p \not\models^{\mathtt{b}} \mathtt{a}'$ for every $\mathtt{a}' \in Ls(\mathtt{a})$. Again we know that $p \Downarrow$. Let $D_{n?y}$ be the finite set $\{p' \mid p \overset{n?y}{\Rightarrow} p'\}$. For each $p' \in D_{n?y}$ we know that if $\mathtt{b}' \ll \mathtt{b}$ then $p' \not\models^{\mathtt{b}} \mathtt{b}'$ and therefore by induction some finite set $J_{p'}$ exists such that $(p' \parallel t(\mathtt{b}')^{J_{p'}}_{X \cup \{y\}})\surd$. The required J is then $\cup\{J_{p'} \mid p' \in D_{n?y}\}$. Note that when $D_{n?y}$ is empty J is also empty.

  Conversely suppose that $(p \parallel t(\mathtt{a})^J_X)\surd$ for some J. Suppose $\mathtt{a}' \ll \mathtt{a}$. We must show that $p \not\models^{\mathtt{b}} \mathtt{a}'$. This is straightforward if $\mathtt{a}'$ is Ω since $(p \parallel t(\mathtt{a})^J_X)\surd$ guarantees that $p \Downarrow$. So we can assume that $\mathtt{a}'$ has the form $\mathtt{n?y.b}'$ where $\mathtt{b}' \ll \mathtt{b}$. In this case it is sufficient to show that $p' \not\models^{\mathtt{b}} \mathtt{b}'$ whenever $p \overset{n?y}{\Rightarrow} p'$. However this is guaranteed by induction since $(p \parallel t(\mathtt{a})^J_X) \xrightarrow{\tau}{}^* p' \parallel t(\mathtt{b}')^J_{X \cup \{y\}}$

- $\mathtt{a} = v(\mathtt{y})\mathtt{n}?\mathtt{y}.\mathtt{b}$: First suppose that $p \not\models^{\mathtt{b}} \mathtt{a}'$ whenever $\mathtt{a}' \ll \mathtt{a}$. If $p \xrightarrow{v(z)n!z} p'$ for no z, p then the required J is the empty set.

  So suppose $p \xrightarrow{v(z)n!z} p'$ where $z \notin fv(\mathtt{a})$. Then $p' \not\models^{\mathtt{b}} \mathtt{b}'$ for every $\mathtt{b}'$ such that $\mathtt{b}'[y/z]$. For if $p' \models^{\mathtt{b}} \mathtt{b}'$ for such a $\mathtt{b}'$ then $p \models^{\mathtt{b}} v\ (\mathtt{z})\mathtt{n}!\mathtt{z}.\mathtt{b}'$ which contradicts the hypothesis since $v\ (\mathtt{z})\mathtt{n}!\mathtt{z}.\mathtt{b}' \ll v\ (\mathtt{y})\mathtt{n}!\mathtt{y}.\mathtt{b}$.

  So we can apply induction to find some J such that $(p' \parallel t(\mathtt{b})^{J}_{X\cup\{z\}})\surd$. We show that $(p \parallel t(\mathtt{b})^{J[y/z]}_{X})\surd$ for this choice of J.

  The only significant move from this term is a communication via a bound output to $v(m)p'[m/y] \parallel ((t(\mathtt{b}^{J[y/z]}_{X\cup\{y\}})[m/y])$. Using the first part of Lemma 5.21 this is the same as $v(m)p'[m/y] \parallel t(\mathtt{b}[m/z])^{J[m/z]}_{X\cup\{m\}})$. Further Lemma 5.19 can be used to show that $q\surd$ implies $(q\sigma)\surd$ whenever σ is an injective substitution since $(q\sigma)\sigma^{-1}$ is the same as q, up to α-conversion. Since $[m/z]$ is an injective substitution it follows from $(p' \parallel t(\mathtt{b})^{J}_{X\cup\{z\}})\surd$ that $(p' \parallel t(\mathtt{b})^{J}_{X\cup\{z\}})[m/z]\surd$, i.e. $(p'[m/y] \parallel t(\mathtt{b}[m/z])^{J[m/z]}_{X\cup\{m\}})\surd$ and therefore $(v(m)p'[m/y] \parallel t(\mathtt{b}[m/z])^{J[m/z]}_{X\cup\{m\}})\surd$.

  The converse is very similar and is left to the reader. □

Corollary 5.23. *$p \sqsubseteq_{\text{must}} q$ implies $\mathtt{AccTr}^{\mathtt{b}}(p) \ll \mathtt{AccTr}^{\mathtt{b}}(q)$.*

Proposition 5.24. *If $\mathtt{AccTr}^{\mathtt{b}}(p) \ll \mathtt{AccTr}^{\mathtt{b}}(q)$ then $p \sqsubseteq_{\text{must}} q$.*

Proof. The proof has the same structure in [10, Lemma 4.4.13], but the details are more complicated because of the different forms of communication allowed in the π-calculus. Suppose $p \ll q$ and p *must* e. We must show q *must* e by proving that for every computation

$$e \parallel q = r_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} r_k \xrightarrow{\tau} \cdots \qquad (*)$$

there exists some i such that $r_i \xrightarrow{\omega}$.

First suppose that this computation is finite, i.e. $r_k \not\xrightarrow{\tau}$ for some k. Each r_i has the form $v(\underline{v}_i)r_i$ where the individual names in $\underline{v}_i$ arise because of the possible exchange of private names between the two processes. By concentrating on the interaction between the two processes, and using Lemma 5.20 this computation can be decomposed into two derivations giving the individual contributions from e, q, respectively:

$$e = e_0 \overset{\bar{a}_1}{\Rightarrow} \cdots e_j \cdots \overset{\bar{a}_k}{\Rightarrow} e_m$$

and

$$q = q_0 \overset{a_1}{\Rightarrow} \cdots q_j \cdots \overset{a_k}{\Rightarrow} q_m$$

These are such that for each j there is some i such that $r_i = e_j \parallel q_j$ and the pair a_i, $\bar{a}_i$ take one of the forms

- $n?x$, $n!x$,
- $n!x$, $n?x$,

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

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 $r_k \stackrel{\tau}{\not\rightarrow}$ we also know that $S(q_m) \cap \overline{S(e_m)} = \emptyset$, where $\bar{S} = \{n? \mid n! \in S\} \cup \{n! \mid n? \in S\}$. For convenience let s denote the sequence of actions a_i, S the set of indications $S(q_m)$ and $\mathtt{a}$ the *acceptance trace* $\mathtt{sS}$. Since $p \ll q$ this means that $p \models^{\mathtt{b}} \mathtt{b}$ for some $\mathtt{b} \ll \mathtt{a}$.

There are now two cases to consider:

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

$$e \parallel \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} e_m \parallel p'$$

The relevant features of this computation are firstly that $e_m \parallel p' \stackrel{\tau}{\not\rightarrow}$ and secondly it only uses e_i which occur in the original computation $(*)$. It follows from p *must* e that $e_i \stackrel{\omega}{\rightarrow}$ 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 $p \stackrel{s'}{\Rightarrow} p'$ where $p \Uparrow$.

We have not touched on the case when 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-conversion, the computation tree from any process term, with respect to the reduction relation $\stackrel{\tau}{\rightarrow}$, is finite branching. □

Combining these two propositions gives the behavioural alternative characterisation, Theorem 5.3:

Theorem 5.25. *$p \sqsubseteq_{\text{must}} q$ if and only if $p \ll q$.*

6. The may case

In this section we briefly outline another Interpretation and show that it is fully abstract with respect to the behavioural preorder $\sqsubseteq_{\text{may}}$. We can make a slight modification to the equation in Section 4,

$$\mathbf{P} \cong \mathbf{A} \times (\mathbf{N} \otimes_{\perp} \mathbf{F}) \times (\mathbf{N} \otimes_{\perp} \mathbf{C})$$
$$\mathbf{F} \cong \mathbf{N} \Rightarrow \mathbf{P}$$
$$\mathbf{C} \cong (\mathbf{N} \otimes \mathbf{P}) + \mathbf{s}(\mathbf{P})$$

but here we solve it in the category $u\mathscr{D}$, i.e. the category whose objects are functors from I to $u\mathscr{SL}$, the category of upper semi-lattices. Note that the only difference is that we have replaced $\top$ with $\perp$ and dropped the use of the lifting functor $(\,)_{\perp}$; the latter is necessary because every upper choice predomain is automatically a domain and

therefore there is no need to add a least element. Moreover, because of the degeneracy of much of the algebraic structure of upper choice domains there is some redundancy in this equation and it can be considerably simplified.

Here **A** denotes the function which at s gives the free upper choice predomain over $s+s$. This is easily shown to be the set of subsets of $s+s$ ordered by set inclusion. One can also show that in the category $u\mathscr{D}$ the action of the functor $(\mathbf{N}\otimes_{\perp} A)$ at s, for any A, is described by the set of partial functions from s to A_s, $s \rightharpoonup A_s$, this time ordered in the more usual fashion: $f \leqslant g$ if $domain(f) \subseteq domain(g)$ and $f(n) \leqslant g(n)$ for every $n \in domain(f)$. Thus instead of using **P** as described above, we could use the simpler equation

$$\mathbf{Q} \cong (\mathbf{N}\otimes_{\perp}\mathbf{F}) \times (\mathbf{N}\otimes_{\perp}\mathbf{C})$$
$$\mathbf{F} \cong \mathbf{N} \Rightarrow \mathbf{Q}$$
$$\mathbf{C} \cong (\mathbf{N}\otimes\mathbf{Q}) + \mathbf{s}(\mathbf{Q})$$

Note that **Q** is not isomorphic to **P**, as objects in $u\mathscr{D}$. But **Q** can be embedded in **P** and is sufficiently rich to support the interpretation of all terms in the language.

To interpret the language in this new domain **P** we proceed exactly as in Section 4; we require an interpretation for each of the language constructs. However, these also can be borrowed directly from Section 4 by making very slight modifications. In this way, we have a new interpretation $[\![\Gamma \vdash t : P]\!]^u$ for each typing judgement and consequently for each process term a family of interpretations $[\![p]\!]^u_s$ from ENV_s to $\mathbf{P}_s$.

Theorem 6.1. *$[\![p]\!]^u_s\rho \leqslant [\![q]\!]^u_s\rho$ if and only if $p\rho \sqsubseteq_{\text{may}} q\rho$.*

This has the immediate corollary:

Corollary 6.2. *For all process terms p, q, $[\![p]\!]^u \leqslant [\![q]\!]^u$ if and only if $p\rho \sqsubseteq_{\text{may}} q\rho$ for every environment ρ.*

The proof of the theorem is very similar to that of Theorem 5.1, although considerably simpler, and here we merely outline the necessary steps.

Let `Seq` be the subset of `AccTr` consisting of all those elements of the form $s\Omega$. The method of associating process terms with these *sequences*, both behaviourally and denotationally, is slightly different than that used in Section 5. Behaviourally, the change is in the clause:

$$p \models^{\mathsf{b}} \Omega \quad \text{for every } p$$

and the denotational change is similar:

$$d \models_s \Omega \quad \text{for every } d \in \mathbf{P}_s.$$

The proof of the behavioural alternative characterisation, Theorem 5.3, can be mimiced to give:

Proposition 6.3. *$p \sqsubseteq_{\text{may}} q$ if and only if $\mathtt{Seq}^{\mathsf{b}}(p) \subseteq \mathtt{Seq}^{\mathsf{b}}(q)$.*

The internal full abstraction result, Theorem 5.3, can also be repeated:

Proposition 6.4. *For all $d, e \in \mathbf{P}_s$, $d \leqslant e$ if and only if $\mathtt{Seq}_{\text{sem}}(d) \leqslant \mathtt{Seq}_{\text{sem}}(e)$.*

The final ingredient is the result corresponding to Theorem 5.5:

Proposition 6.5. *If $\underline{x} \vdash p : P$, $x_i \in s$ for every i and $id \in ENV_s$ is the identity on s then $\mathtt{Seq}^{\mathsf{b}}(p) = \mathtt{Seq}(\llbracket p \rrbracket^u_s id)$.*

These three results can be combined, as in Section 5, to give a proof for Theorem 6.1.

We end this section and the paper with a brief look at how these models can be used to characterise more general contextual forms of testing. As an illustration we use *may testing*; corresponding theorems for the must case are still open.

Let us write $p \Downarrow_{\text{may}}$ if $p \xrightarrow{\tau} q^*$ for some q which can perform some communication, i.e. $q \xrightarrow{n?m}$ or $q \xrightarrow{n!m}$ for some n, m.

Definition 3. Let $p \sqsubseteq_{\text{mayconv}} q$ if $C[p] \Downarrow_{\text{may}}$ implies $C[q] \Downarrow_{\text{may}}$ for every context $C[\,]$.

This is the form of testing used in, for example [5], and is a natural generalisation of that used for the λ-calculus in [2].

Theorem 6.3. *Theorem 6.6 $p \sqsubseteq_{\text{mayconv}} q$ if and only if $\llbracket p \rrbracket \leqslant \llbracket q \rrbracket$.*

Proof (*Outline*)**.** First suppose $\llbracket p \rrbracket \leqslant \llbracket q \rrbracket$ and $C[p] \Downarrow_{\text{may}}$. Then by the construction of the model $\llbracket C[p] \rrbracket \leqslant \llbracket C[q] \rrbracket$. Let s be a sufficiently large set of names which includes all those free in both $C[p]$ and $C[q]$. Then we have $\llbracket C[p] \rrbracket_s id \leqslant C[q]_s id$ and from Proposition 6.5 this means $\mathtt{Seq}^{\mathsf{b}}(C[p]) \subseteq \mathtt{Seq}^{\mathsf{b}}(C[q])$. In general $r \Downarrow_{\text{may}}$ if and only if $\mathtt{Seq}(r)$ is not $\{\varepsilon\}$, and therefore it follows that $C[q] \Downarrow_{\text{may}}$.

Conversely suppose that for all contexts $C[\,]$, $C[p] \Downarrow_{\text{may}}$ implies $C[q] \Downarrow_{\text{may}}$. We must show $\llbracket p \rrbracket \leqslant \llbracket q \rrbracket$. For this it is sufficient to show that $\llbracket p \rrbracket_s \rho \leqslant \llbracket q \rrbracket_s \rho$ for every s and for every environment ρ. From the full abstraction result for *may* testing it is sufficient in turn to show that for every process e that $p\rho$ *may* e implies $q\rho$ *may* e. However it is easy to simulate *may* testing in terms of testing for $\Downarrow_{\text{may}}$ in arbitrary contexts and substitutions can be simulated using input prefixing. For example the effect of the substitution of x for y in r can be obtained by the context $\nu(n)(n?\lambda y.p \| n!x.0)$. □

Appendix A. Definition of parallel

The natural transformation $\|_{\mathbf{P}} : \mathbf{P} \otimes \mathbf{P} \dot{\to} \mathbf{P}$ is defined to be $\mathbf{par}^{\otimes}$, where $\mathbf{par} : (\mathbf{P} \times \mathbf{P}) \dot{\to} \mathbf{P}$ is a natural transformation which is linear in both its arguments, and

is defined, in analogy with the interleaving law, in terms of two auxiliary natural transformations, **ext**, **int**, by

$$(+_{\mathbf{P}} \circ \langle \mathbf{ext} \times \mathbf{int} \rangle) \wedge \mathbf{int}$$

These natural transformations are in turn defined in terms of **par** and therefore in fact we take **par** to be the least fixpoint of its definition.

To define **ext**, **int** we need to develop some notation. For any $\tau : \mathbf{P} \times \mathbf{P} \dot{\to} \mathbf{P}$ let $\tau_{\mathbf{F}} : (\mathbf{N} \Rightarrow \mathbf{P}) \times \mathbf{P} \dot{\to} (\mathbf{N} \Rightarrow \mathbf{P})$ be the natural transformation defined by letting $(\tau_{\mathbf{F}})_s : (\mathbf{N} \Rightarrow \mathbf{P})_s \times \mathbf{P}_s \to (\mathbf{N} \Rightarrow \mathbf{P})_s$ be the morphism

$$(\tau_{\mathbf{F}})_s(f, d) = \lambda x \in \mathcal{N}.\tau_{s \cup \{x\}}(f(x), d).$$

We can also define a natural transformations $\tau_{\mathbf{C}} : \mathbf{C} \times \mathbf{P} \dot{\to} \mathbf{C}$ by letting $(\tau_{\mathbf{C}})_s(f, d)$ be the element in $\mathbf{C}$ defined by $\lambda x \in domain(f).\tau_{s \cup \{x\}}(f(x), d)$. Any natural transformation $\tau : (A \times B) \dot{\to} C$ can be extended in an obvious way to a natural transformation in $(\mathbf{N} \otimes^{\top} A \times B) \dot{\to} \mathbf{N} \otimes^{\top} C$ and let us use the non-standard notation $\mathbf{N} \otimes^{\top} \tau$ for this extension.

Then the natural transformation $\mathbf{ext} : (\mathbf{P} \times \mathbf{P}) \dot{\to} \mathbf{P}$ is given by $(\mathbf{par_A} \times \mathbf{par_F} \times \mathbf{par_C})_{\perp}$ where

- $\mathbf{par_A}$ is $+_{\mathbf{A}} \circ \langle \pi_1 \circ down \circ \pi_1 \times \pi_1 \circ down \circ \pi_2 \rangle$,
- $\mathbf{par_F}$ is

$$((\mathbf{N} \otimes^{\top} \mathbf{par_F}) \circ \langle \pi_2 \circ down \circ \pi_1 \times \pi_2 \rangle) \wedge ((\mathbf{N} \otimes^{\top} \mathbf{par_F}) \circ \langle \pi_2 \circ down \circ \pi_2 \times \pi_1 \rangle)$$

- $\mathbf{par_C}$ is

$$((\mathbf{N} \otimes^{\top} \mathbf{par_C}) \circ \langle \pi_3 \circ down \circ \pi_1 \times \pi_2 \rangle) \wedge ((\mathbf{N} \otimes^{\top} \mathbf{par_C}) \circ \langle \pi_3 \circ down \circ \pi_2 \times \pi_1 \rangle)$$

Intuitively, $\mathbf{ext}_s(d, e)$ describes the element of $\mathbf{P}_s$ obtained by interleaving the possible actions of the elements d, e of $\mathbf{P}_s$ without taking into consideration possible *communications* between them.

These *communications* are the concern of the second natural transformation **int**. For any $\tau : (\mathbf{P} \times \mathbf{P}) \dot{\to} \mathbf{P}$ let $\tau_{\mathbf{FC}} : (\mathbf{F} \times \mathbf{C}) \dot{\to} \mathbf{P}$ be defined by letting $(\tau_{\mathbf{FC}})_s : (\mathbf{F}_s \times \mathbf{C}_s) \to \mathbf{P}_s$ be the morphism defined by

$$(\tau_{\mathbf{FC}})_s(f, g) = \sum \{\tau_{s \cup \{n\}}(f(n), g(n)) \mid n \in domain(g)\}$$

This is now generalised to a natural transformation

$$\tau^{\mathbf{N}}_{\mathbf{FC}} : (\mathbf{N} \otimes^{\top} \mathbf{F}) \times (\mathbf{N} \otimes^{\top} \mathbf{C},) \dot{\to} \mathbf{P}^{\top}$$

by the definition

$$(\tau^{\mathbf{N}}_{\mathbf{FC}})_s(g_1, g_2) = \begin{cases} \sum \{(\tau_{\mathbf{FC}})_s(g_1 n, g_2 n) \mid n \in s'\}, & s' \neq \emptyset \\ \top, & s' = \emptyset \end{cases}$$

where $s' = domain(g_1) \cap domain(g_2)$. Note that the codomain of this natural transformation needs to be $\mathbf{P}^{\top}$ rather than $\mathbf{P}$.

Finally, the natural transformation $\mathbf{int}: (\mathbf{P} \times \mathbf{P}) \dot{\rightarrow} \mathbf{P}^{\top}$ is defined by

$$(\tau^{\mathbf{N}}_{\mathbf{FC}} \circ \langle \pi_2 \circ \pi_1 \times \pi_3 \circ \pi_2 \rangle) \wedge (\tau^{\mathbf{N}}_{\mathbf{FC}} \circ \langle \pi_2 \circ \pi_2 \times \pi_3 \circ \pi_1 \rangle)$$

Acknowledgements

The author would like to thank Cosimo Laneve, Dusko Pavlovic, Davide Sangiori and anonymous referees for useful comments on earlier drafts of this paper.

References

[1] S. Abramsky, A domain equation for bisimulation, Inform. and Comput. 92 (1991) 161–218.
[2] S. Abramsky, C. Ong, Full abstraction in the lazy lambda calculus, Inform. and Comput. 105 (1989) 159–267.
[3] H. Barendregt, The Lambda Calculus Studies in Logic, vol. 103, North-Holland, Amsterdam, 1984.
[4] M. Boreale, R. DeNicola, Testing for mobile processes, Inform. and Comput. 120 (1995) 279–303.
[5] G. Boudol, A lambda calculus for (strict) parallel functions, Inform. and Comput. 108 (1994) 51–127.
[6] R. Crole, Categories for Types, Cambridge University Press, Cambridge, 1993.
[7] R. DeNicola, M. Hennessy, Testing equivalences for processes, Theoret. Comput. Sci. 24 (1984) 83–113.
[8] F.J. Oles, Type algebras, functor categories and block structure, in: M. Nivat, J. Reynolds (Eds.), Algebraic Methods in Semantics, Cambridge University Press, Cambridge, 1985, pp. 543–573.
[9] C. Gunter, Semantics of Programming Languages, MIT Press, Cambridge, MA, 1992.
[10] M. Hennessy, An Algebraic Theory of Processes, MIT Press, Cambridge, MA, 1988.
[11] M. Hennessy, A Model for the π Calculus, Tech. Report 8/91, University of Sussex, 1991.
[12] M. Hennessy, A. Ingolfsdottir, A theory of communicating processes with value-passing, Inform. and Comput. 107 (2) (1993) 202–236.
[13] D.J. Howe, Equality in lazy computation systems. Proc. 4th IEEE Symp. on Logic in Computer Science, 1989 pp. 198–203.
[14] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[15] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Part i, Inform. and Comput. 100 (1) (1992) 1–40.
[16] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Part ii, Inform. and Comput. 100 (1) (1992) 41–77.
[17] R. Milner, The polyadic π-calculus: a tutorial, Proc. Internat. Summer School on Logic and Algebra of Specification, Marktoberdorf, 1991.
[18] E. Moggi, Notions of computation and monad, Inform. and Comput. 93 (1991) 55–92.
[19] E. Moggi, M.P. Fiore, D. Sangiorgi, A fully-abstract model for the π-calculus Proc. LICS'96, 1996.
[20] P.W. O'Hearn, R.D. Tennant, Semantics of local variables. Applications of Categories in Computer Science London Mathematical Society Lecture Note Series, Cambridge University Press, Cambridge, 1992 pp. 217–238.
[21] J. Parrow, D. Sangiorgi, Algebraic theories for name-passing calculi, Inform. and Comput. 120 (1995) 174–197.
[22] A.M. Pitts, I. Stark, On the observable properties of higher order functions that dynamically create local names, or: what's new, Proc. Mathematical Foundations of Computer Science, Springer, Berlin, 1993, pp. 122–141.
[23] D. Sangiorgi, Expressing mobility in process algebras: first-order and higher-order paradigms, Ph.D. Thesis, Edinburgh University, Scotland, 1992.
[24] D. Sangiorgi, A theory of bisimulation for the π-calculus, Proc. CONCUR'93, Lecture Notes in Computer Science, vol. 715 in 1993.
[25] I. Stark, A fully-abstract domain model for the π-calculus, Proc. LICS'96, 1996.