A Fully Abstract Denotational Model for Higher–Order Processes*

M. Hennessy
University of Sussex

Abstract

A higher–order process calculus is defined in which one can describe processes which transmit as messages other processes; it may be viewed as a generalisation of the lazy λ-calculus. We present a denotational model for the language, obtained by generalising the domain equation for Abramskys model of the lazy λ-calculus. It is shown to be fully abstract with respect to three different behavioural preorders. The first is based on observing the ability of processes to perform an action in all contexts, the second on *testing* and the final one on satisfying certain kinds of modal formulae.

*This work has been supported by the SERC grant GR/H16537

1 Introduction

Process algebras are simple specification languages for concurrent communicating processes. Typically they consist of a small number of combinators for constructing new processes from existing processes and their meaning is then determined by a collection of laws or equations expressed in terms of these combinators. For example CCS, [Mil89], contains a parallel and a choice combinator, $|$ and $+$ respectively. The term $p \mid q$ describes a process which consists of two subprocesses p and q running in parallel while $p + q$ describes a process which may either act like p or like q but not both. It also contains a set of prefixing combinators, one for each *action* from some predefined action set. The term $c.p$ is a process which can perform the action c and then proceed to act like the process p. These actions may be interpreted in a variety of ways but typically they represent the sending or receipt of data along some communication channel and communication is modelled by the simultaneous occurrence of a send and a receive. Combinators similar in style to these appear in most process algebras as does some form of scoping for channel names. Indeed it is this last concept which gives them much of their descriptive power.

The underlying mathematical theory of these languages is well-developed and fairly well understood, [Mil89, Hoa85, BW90, Hen88]. Much of this fundamental work has been carried out for "pure" process algebras, where the actions are taken to be simple synchronisation pulses along channels, but more recently theories have been developed for languages where various kinds of data are passed along the communication channels. For example in [HI91] simple data values such as the integers are allowed while in [MPW92a, MPW92b] channels themselves are allowed. In [Tho89, Tho90] processes may pass other processes as values and it is this type of process description language which is the topic of the present paper.

There are now two kinds of prefixing, $c?X.P$, meaning input a process along the channel c and bind it to the process variable X in the term P, and $c!Q.P$, meaning output the process Q along the channel c and then act like the process P. Thus $c?X.(X \mid R)$ represents a process which can input any process and run it in parallel with R. So combining this with $c!Q.P$ we obtain the process $c!Q.P \mid c?X.(X \mid R)$ which can perform a communication to become the process $P \mid (Q \mid R)$. This idea is pursued in depth in [Tho90] where a number of different formalisations are investigated. The resulting language is shown to be very powerful in that it can simulate, in some sense, both the λ-calculus and the π-calculus of [MPW92a]. The connection between the π-calculus and various *higher–order* process calculi and their relative merits is further pursued in [San92]. Here we do not address such issues. Rather we investigate the possibility of providing an adequate semantic theory for higher–order process calculi. In particular we provide a fully abstract denotational model for one such higher–order language.

The starting point for the development of this model is the lazy λ-calculus. At a very naive level this is a primitive higher-order process language. The λ-term $\lambda x.p$ may be viewed as a process which is waiting to receive another process along the communication channel λ while the application term pq represents sending the process q to the process p. In [Abr90] this language is interpreted in the model obtained by solving the domain equation

$$\mathbf{D} \;=\; \mathbf{F}_{\perp}$$

$$\mathbf{F} \;=\; \mathbf{D} \longrightarrow \mathbf{D}$$

Each λ-term is interpreted either as $\bot$, in the case when it gives rise to a divergent computation, or as an element of $\mathbf{F}$, i.e. a function over λ-terms. A higher-order process can be viewed as having similar behaviour but now parametrised on channels; λ-terms being simple processes which can only receive input on one channel. Thus the input behaviour of a higher–order process, in analogy with λ-terms, can be captured by a function from $\mathcal{N}$, the set of channel names, to $\mathbf{F}_\bot$; with respect to each channel the process may offer no behaviour, modelled by $\bot$, or may act like a function over processes. Similarly its output behaviour, which has no real counterpart in the λ-calculus, can be modelled as a function from $\mathcal{N}$ to $\mathbf{C}_\bot$, where $\mathbf{C}$ is some space suitable for modelling output. One simple suggestion for $\mathbf{C}$ is the Cartesian product $\mathbf{D} \times \mathbf{D}$, with the elements of the pair representing, respectively, the process sent along the channel and the residual of the output action. We shall see that a slightly more complicated form of product is actually necessary, which we denote by $\mathbf{C} \otimes^r \mathbf{C}$.

The analogy between λ-terms and higher–order processes given above is rather tenuous but it has helped us motivate a model obtained by solving the recursive domain equation

$$\begin{aligned}
\mathbf{P} &= (\mathcal{N} \longrightarrow \mathbf{C}_\bot) \times (\mathcal{N} \longrightarrow \mathbf{F}_\bot) \\
\mathbf{F} &= \mathbf{P} \longrightarrow \mathbf{P} \\
\mathbf{C} &= \mathbf{P} \otimes^r \mathbf{P}
\end{aligned}$$

Moreover one can easily imagine how the combinators of input and output might be interpreted over this model. But a reasonable process algebra contains other combinators. At first glance the choice combinator $+$ would appear to present problems. The λ-calculus is completely deterministic in its extensional behaviour but nondeterminism is an essential aspect of process algebras. However in [Bou91] it is shown how to interpret a nondeterministic version of the λ-calculus in the domain $\mathbf{D}$; this is in fact a complete lattice and in particular it has a join operator $\vee$. The domain $\mathbf{P}$ is also a complete lattice and thus we can use exactly the same approach, interpreting the choice combinator $+$ as $\vee$. Moreover having an interpretation of prefixing and choice means that by means of an expansion theorem we can obtain an interpretation of the parallel combinator $|$. Finally it turns out that we can also interpret in a straightforward fashion certain forms of channel scoping; that called *dynamic scoping* in [Tho90].

Having outlined a possible denotational semantics for a higher–order process calculus the next question we must address is: how reasonable is this as a model of the behaviour of processes? To answer this question we again pursue the analogy with the λ-calculus. How good is $\mathbf{D}$ as a model of the behaviour of λ-terms ? This question is answered in a very precise manner in [Abr90, Bou91] and we can give a very similar answer for $\mathbf{P}$.

In [Abr90] an observational preorder is defined on λ-terms in terms of their ability to converge to a "functional term" of the form $\lambda x.r$ using a *lazy* evaluation strategy. This is a variation on the original observational preorder on λ-terms defined by Morris, see exercise 16.5.5 of [Bar84]. Let us denote this by $p \Downarrow$ and for two λ-terms p, q let $p \prec q$ if $p \Downarrow$ implies $q \Downarrow$.

> Let $p \;\underline{\sqsubseteq}_{\mathcal{O}}\; q$ if for every context $C[\ ]$, i.e. a term with one "hole" $[\ ]$ in it, $C[p] \prec C[q]$.

In [Abr90] it is shown that, subject to certain expressivity requirements, the domain $\mathbf{D}$ is fully abstract with respect to the observational preorder $\precsim_{\mathcal{O}}$. That is, $p \precsim_{\mathcal{O}} q$ if and only if the interpretation of p in the domain $\mathbf{D}$ is dominated by the interpretation of q; the domain properly reflects the ability of λ-terms to act like functions. A similar result holds for the the nondeterministic or parallel version of the λ-calculus of [Bou90a, Bou91] but $p \Downarrow$ is interpreted as *it is possible for* p to converge to a functional term, although in these papers a different phraseology is used.

Viewing the λ-calculus as a primitive higher–order process calculus $p \Downarrow$ can be interpreted as: p is willing to offer a communication on the communication channel λ. So let us generalise this predicate $\Downarrow$ to arbitrary processes from our higher–order process calculus by defining

$p \Downarrow$ if there exists some channel on which p is willing to offer a communication

The main result of this paper is that, subject once more to expressivity requirements, the model $\mathbf{P}$ is fully-abstract with respect to the observational preorder $\precsim_{\mathcal{O}}$, with this new interpretation of $\Downarrow$. That is, the interpretation of the process p in the domain $\mathbf{P}$ is dominated by that of q if and only if for every context $C[\ ]$ if $C[p]$ is willing to offer a communication on some channel then so is $C[q]$.

We also prove full abstraction for two other observational preorders between processes and both can also be motivated by reference to similar results for the lazy λ-calculus. The ability to examine a λ-term in an arbitrary context gives one complete control over that term; the context can for example send the term to a collection of subterms each of which can examine an aspect of its behaviour and then pass it on to other subterms for further examination. However each of these subterms can only use the term under examination in a limited manner: they can only supply an argument for the term to be applied to. So a simpler behavioural preorder may be defined on λ- terms based on their reaction to a sequence of arguments:

$p \precsim_{\mathcal{T}} q$ if $(\ldots(pr_1)\ldots r_n) \Downarrow$ implies $(\ldots(qr_1)\ldots r_n) \Downarrow$ for every sequence of λ-terms $r_1, \ldots, r_n$.

The model $\mathbf{D}$ is also fully abstract with respect to this preorder, i.e. $\precsim_{\mathcal{O}}$ and $\precsim_{\mathcal{T}}$ coincide over λ-terms. This view of λ-terms treats them as "black boxes". One has no control over them; the only way of finding out about their behaviour is to send them a parameter, i.e. communicate with them. This is very similar in spirit to the theory of testing for processes, originally presented in [DH84] and expounded at length in [Hen88]. There a test e (represented as another process) is applied to a process p by running e and p in parallel, thereby allowing them to communicate, and the application is successful if e reaches some "successful" state. The test e may be viewed as a generalisation of the sequence of parameters $r_1, \ldots r_n$ supplied to the λ-term and the successful state plays the role of $\Downarrow$. So let us generalise $\precsim_{\mathcal{T}}$ to higher-order processes by saying p *may satisfy* the test e if there is a successful application of e to p and

$p \precsim_{\mathcal{T}} q$ if p *may satisfy* e implies q *may satisfy* e for every test e.

We show that $\mathbf{P}$ is also fully abstract with respect to $\precsim_{\mathcal{T}}$.

The full abstraction results in [Abr90, Bou91] rely heavily on a logical characterisation of the domain $\mathbf{D}$, [CC90, Abr91]. Essentially the compact elements of $\mathbf{D}$ can be described

by formulae from a logic in such a way that $\mathbf{D}$ is isomorphic to the filters generated by the logic. Further the interpretation of the λ-calculus in $\mathbf{D}$ can be completely captured by a program logic whose judgements are of the form $\vdash p : \phi$ and this program logic is central to the proofs of full abstraction. A similar program can be carried out for the model $\mathbf{P}$ and the resulting logic is a simple modal logic whose formulae express the ability of processes to receive and transmit along communication channels. Furthermore we can interpret this modal language operationally over processes using a satisfaction relation $\models$ between processes and formulae; it is called a realizability interpretation in [Bou91]. This gives a further method for comparing behaviourally processes:

$$\text{let } p \ \underset{\sim}{\sqsubset}_{\mathcal{L}} \ q \text{ if } p \models \phi \text{ implies } q \models \phi \text{ for every formulae } \phi$$

We also show that $\mathbf{P}$ is fully abstract with respect to $\underset{\sim}{\sqsubset}_{\mathcal{L}}$. This is analogous to the modal characterisation of bisimulation equivalence for processes given in [HM85].
To sum up the three behavioural preorders $\underset{\sim}{\sqsubset}_{\mathcal{O}}$, $\underset{\sim}{\sqsubset}_{\mathcal{T}}$ and $\underset{\sim}{\sqsubset}_{\mathcal{L}}$ coincide over higher–order processes definable in our language and further they are characterised precisely by the model $\mathbf{P}$.

We now describe in some detail the contents of each section of the paper. The next section is a review of the mathematical properties of the domains which we use, prime algebraic lattices. In section three the language for higher order processes is given together with its operational semantics. The syntax is different from that used in [Tho90] as we borrow some syntactic constructs from [Mil91], namely *abstractions* and *concretions*.

The language is typed, having terms of type process, abstraction and concretion. The input construct $c?X.T$ is replaced by the term of type process $c?F$ where F is of type abstraction. For simplicity we restrict abstraction terms to have the form $(X)T$ where T is of type process although this could easily be generalised. Thus $c?X.T$ is replaced by $c?(X)T$ and the term $(X)T$ can be seen as a functional abstraction and will indeed be interpreted in much the same way as the λ-term $\lambda X.T$. Similarly the output construct $c!T.U$ is replaced by $c!C$ where C is a term of type *concretion*. All concretion terms have the form $[R]S$ where R and S have type process; R is the process sent along the channel c while S represents the subsequent behaviour of $c![R]S$ after the output has been performed. The language also contains the standard choice operator $+$ and a generalised parallel construct $_{\mathcal{A}|\mathcal{B}}$ over process terms where $\mathcal{A}$, $\mathcal{B}$ are any subsets of the of channels $\mathcal{N}$. Intuitively in $P \ _{\mathcal{A}|\mathcal{B}} \ Q$ $P$and Q can communicate at will using common channels but P may only communicate with processes other than Q using channels from $\mathcal{A}$ and similarly for Q and $\mathcal{B}$. This generalisation of the usual parallel construct automatically incorporates a form of channel scoping. The usual channel restriction combinator $P\backslash\mathcal{A}$, where P may not use the channels from $\mathcal{A}$ for external communication, can be implemented as $P \ _{\mathcal{B}|\mathcal{N}} \ NIL$ where NIL is the usual constant representing the "empty" process and $\mathcal{B}$ is $\mathcal{N} - \mathcal{A}$. Our language may be viewed as an extension of CHOCS, [Tho90], and therefore we refer the reader to that publication in order to get some idea of its expressiveness.

Section three also contains the operational semantics. This is expressed in terms of three judgements:

- $P \xrightarrow{\tau} Q$: This means that the process P may perform a communication and thereby be transformed into Q. This is in turn defined in terms of the two other judgements.

- $P \xrightarrow{n!} C$: This formalises the fact that P is willing to send a communication to the channel n and C encodes a possible result of that action

- $P \xrightarrow{n?} F$: Here P is willing to receive a process from the channel n and F encodes what will be done with any such process received.

Finally section three also contains formal definitions of the three behavioural relations $\sqsubseteq_{\mathcal{O}}$, $\sqsubseteq_{\mathcal{T}}$ and $\sqsubseteq_{\mathcal{L}}$, all defined in terms of the operational semantics.

The denotational model $\mathbf{P}$ is defined in section four and its properties elaborated. The main result is its characterisation in terms of a modal logic.

The next two sections are devoted to the semantic interpretation of the language. This is carried out in two ways. The first is a fairly standard denotational semantics in that each combinator is interpreted as a function of the appropriate type over the domains. The second consists of a program logic with judgements of the form $\vdash P : \phi$ where P is a process and ϕ a modal formulae or more generally $\Gamma \vdash T : \phi$ where Γ is a set of assumptions about the free variables in the term T. There are two main results concerning the interpretations. The first shows that the two interpretations are essentially equivalent; each can be recovered from the other. This is proved in section six. The second, in section five, is a definability result which says that every compact element of the model is denotable by a term in the language.

The full abstraction results are then given in section six. This is a relatively short section as it merely combines the various results which have been accumulated in the previous sections. The principal technical result is called *adequacy*. This states that the interpretation of a process in the model is equal to $\perp$, the least element of the domain $\mathbf{P}$, if and only if the process can never offer a communication on any channel. This adequacy result together with that on definability are the central components in the proofs of full abstraction. In turn the logical characterisation of the semantic interpretation is crucial to the proof of adequacy.

The paper ends with a short conclusion outlining possible future research and related work.

2 Mathematical Preliminaries

In this section we define our notion of domains and outline some of their properties.

A complete lattice is a partial order $(D, \leq)$ such that each subset of X of D has a least upper bound which we denote by $\bigvee X$. We write the lub of two elements as $x \vee y$. An element c in a complete lattice is *compact* if for each subset X of D such that $c \leq \bigvee X$ there exists a finite subset Y of X such that $c \leq \bigvee Y$. Let $\mathcal{K}(D)$ denote the compact elements of the complete lattice D. A complete lattice D is *algebraic* if for every $d \in D$

$$d = \bigvee \{\, c \in \mathcal{K}(D) \mid c \leq d \,\}.$$

An compact element p of D is *prime* if $p \leq d \vee e$ implies either $p \leq d$ or $p \leq e$. Let $\mathcal{KP}(D)$ be the set of prime elements of the complete lattice D. Then a lattice is said to

be *prime algebraic* if for every $d \in D$

$$d = \bigvee \{ c \in \mathcal{KP}(D) \mid c \leq d \}.$$

In this paper we use *domain* to mean a prime algebraic lattice. Note that every domain D has a least element $\bot = \bigvee \emptyset$ and a greatest element $\top = \bigvee D$. Also every compact element c is the join of a finite number of primes, $c = p_1 \vee \ldots \vee p_n$.

A function $f: D \longmapsto E$ between two domains is *strict* if $f(\bot) = \bot$, *monotonic* if $d \leq d'$ implies $f(d) \leq f(d')$ and injective if $f(d) \leq f(d')$ implies $d \leq d'$. A subset X of a domain D is *directed* if every finite $Y \subseteq X$ has an upper bound in X, i.e. there exists some d in X such that $\bigvee Y \leq d$. It follows that every directed set is non-empty since it must contain an upper bound for the empty set. The function f is *continuous* if for every directed subset X of D $f(X) = \{ f(x) \mid x \in X \}$ is directed and $f(\bigvee X) = \bigvee f(X)$. Note that continuous functions are automatically monotonic. We say f is *linear* if in addition $f(x \vee y) = f(x) \vee f(y)$ and more generally if $f: D^k \longmapsto E$ then it is *multilinear* if $f(d_1, \ldots, d_i \vee d_i', \ldots d_k) = f(d_1, \ldots, d_i, \ldots d_k) \vee f(d_1, \ldots, d_i', \ldots d_k)$.

Domains are completely determined by their primes, as we shall now see. Moreover continuous functions are determined by their effect on compact elements and multilinear functions by their effect on primes.
For any partial order $(P, \leq)$ with a least element $\bot$ let $Fin(P)$ be the set of finite non-empty subsets of P and for $u, v \in Fin(P)$ let

$$u \leq_l v \text{ if } \forall x \in u \exists y \in v . x \leq y.$$

Then $(Fin(P), \leq_l)$ is a pre-partial order or *ppo*, i.e. $\leq_l$ is reflexive and transitive, and it has a least element, namely $\{\bot\}$. Let $\mathcal{P}_l(P)$ denote its ideal completion, [Gue81]. This is often refered to as the lower or Hoare powerdomain of P.

Lemma 2.1

1. *$\mathcal{P}_l(P)$ is a domain with (the embedding of) $Fin(P)$ as its compact elements and (the embedding of) P as its primes*

2. *Every domain D is isomorphic to $\mathcal{P}_l(\mathcal{KP}(D))$.* □

As a result if $\mathcal{KP}(D)$ and $\mathcal{KP}(E)$ are isomorphic as "partial orders with $\bot$" then D and E are isomorphic as domains.

For any monotonic $f: \mathcal{K}(D) \longmapsto E$ let $f^\infty: D \longmapsto E$ be defined by

$$f^\infty(d) = \bigvee \{ f(c) \mid c \leq d \}$$

and for any $f: D \longmapsto E$ let $f_c: \mathcal{K}(D) \longmapsto E$ be the natural restriction. In a similar manner for any monotonic $f: \mathcal{KP}(D)^k \longmapsto E$ let $f^\infty: D^k \longmapsto E$ be defined by

$$f^\infty(\underline{d}) = \bigvee \{ f(\underline{p}) \mid \underline{p} \leq \underline{d} \}$$

and for $f: D^k \longmapsto E$ let $f_p: \mathcal{KP}(D)^k \longmapsto E$ be the natural restriction.

Lemma 2.2

1. *f is continuous if and only if $f = (f_c)^\infty$*

2. *f is multilinear if and only if $f = (f_p)^\infty$* □

Thus for continuous functions $f = g$ if and only if $f_c = g_c$ and similarly for multilinear functions. It follows that in order to define a continuous (multilinear) function it is sufficient to define it on the compact (prime) elements.

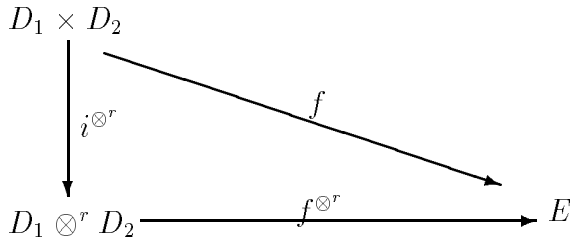
We now review briefly the constructions of domains which are required in the paper; most are standard. For any set N let $(N \longrightarrow E)$ be the set of all functions from N to the domain E. These functions are ordered in the standard way, namely $f \leq g$ if $f(n) \leq g(n)$ for every n in N. With this ordering $(N \longrightarrow E)$ is a domain where the primes are all those functions f whose range is $\mathcal{KP}(E)$ and which return $\perp$ for all but at most one element of N.
Let $[D \longrightarrow E]$ be the set of continuous functions from the domain D to the domain E. This, ordered in the standard way, can also be seen to be a domain where the primes are step functions of the form $c \Rightarrow p$ for $c \in \mathcal{K}(D)$ and $p \in \mathcal{KP}(E)$. Recall that the step function $d \Rightarrow e$ is defined by

$$d \Rightarrow e(x) = \begin{cases} e & d \leq x \\ \perp & \text{otherwise.} \end{cases}$$

The Cartesian product $D \times E$ also yields a domain as does the "lifting operation" $D_\perp$. We use $d_\perp$ to denote the element $in_\perp(d)$ of $D_\perp$ where $in_\perp : D \longmapsto D_\perp$ is the natural injection.

The most complicated construction we require is a form of tensor product. In the Cartesian product $D_1 \times D_2$ the join is defined pointwise: $(d_1, d_2) \vee (d_1', d_2') = (d_1 \vee d_1', d_2 \vee d_2')$. This implies $(d, d_1 \vee d_2) = (d, d_1) \vee (d, d_2)$ and $(d_1 \vee d_2, d) = (d_1, d) \vee (d_2, d)$. To model concretions, as outlined in the introduction, we require a product where the former identity remains true but in general $(d_1 \vee d_2, d)$ is different from $(d_1, d) \vee (d_2, d)$. This is defined in the following way. A continuous function $f : D_1 \times D_2 \longmapsto E$ is called *right-linear* if $f(d_1, d_2 \vee d_2') = f(d_1, d_2) \vee f(d_1, d_2')$. For any two domains D_1, D_2 let $D_1 \otimes^r D_2$ be the domain characterised by the requirements

1. there exists a right-linear injection $i^{\otimes^r} : D_1 \times D_2 \longmapsto D_1 \otimes^r D_2$
2. for any right-linear $f : D_1 \times D_2 \longmapsto E$ there exists a unique linear $f^{\otimes^r} : D_1 \otimes^r D_2 \longmapsto E$ which makes the following diagram commute:



Of course we have to show that such a $D_1 \otimes^r D_2$ exists. A standard "arrow-chasing" argument will establish that if it exists it is unique (up to isomorphism) and we content ourselves with outlining the construction of one domain with both of the required properties.

In fact to construct $D_1 \otimes^r D_2$ it is sufficient to define its prime elements. Let

$$P = \{ (c, p) \mid c \in \mathcal{K}(D_1),\ p \in \mathcal{KP}(D_2) \}$$

and let $(c,p) \leq (c',p')$ if $c \leq_{D_1} c'$ and $p \leq_{D_2} p'$. This is a ppo with a least element and we let $D_1 \otimes^r D_2$ be $\mathcal{P}_l(P)$. Let $i: \mathcal{K}(D_1 \times D_2) \longmapsto D_1 \otimes^r D_2$ be defined by

$$i(c_1,c_2) = \{\, (c_1,p_i) \mid c_2 = p_1 \vee \ldots \vee p_n \,\}$$

where we identify $Fin(P)$ with its injection into $D_1 \otimes^r D_2$. Note that this is well defined for if $p_i,\ q_j$ are all prime and $p_1 \vee \ldots \vee p_k = q_1 \vee \ldots \vee q_l$ then $\{\, (c,p_i) \mid i \in I \,\} =_l \{\, (c,q_j) \mid j \in j \,\}$. One can also check that it is monotonic, right-linear and injective. Let $i^{\otimes^r}$ be defined as i^∞ which is therefore a right–linear injection.

We must check that i^∞, defined in the way, has the required properties. To this end let $f: D_1 \ \times D_2 \longmapsto E$ be an arbitrary right-linear function. Define $f_l: \mathcal{KP}(D_1 \otimes^r D_2) \longmapsto E$ by $f_l(c,p) = f(c,p)$. This is obviously monotonic and therefore we can define $f^{\otimes^r}$ to be function f_l^∞. We must check

1. $f^{\otimes^r} \circ i^{\otimes^r} = f$.
   Again it is sufficient to consider compact elements of $D_1 \times D_2$. Let $c_i \in \mathcal{K}(D_i)$ and let $c_2 = p_1 \vee \ldots \vee p_n$. Then

$$\begin{aligned}
f^{\otimes^r} \circ i^{\otimes^r}(c_1,c_2) &= f^{\otimes^r}(\{\, c_1,p_i \mid 1 \leq i \leq n \,\}) \ \text{by the definition of } i^{\otimes^r} \\
&= \bigvee_i f^{\otimes^r}(c_1,p_i) \ \text{since } f^{\otimes^r} \text{ is linear} \\
&= \bigvee_i f(c_1,p_i) \ \text{by the definition of } f^{\otimes^r} \\
&= f(c_1, \bigvee_i p_i) \ \text{since } f \text{ is right-linear} \\
&= f(c_1,c_2)
\end{aligned}$$

2. If $g: D_1 \otimes^r D_2 \longmapsto E$ is linear and $g \circ i^{\otimes^r} = f$ then $g = f^{\otimes^r}$.
   It is sufficient to check that they coincide on the primes of $D_1 \otimes^r D_2$ which is obvious since they both must coincide with f there.

We will usually use the notation $d_1 \otimes^r d_2$ in place of $i^{\otimes^r}(d_1,d_2)$. Using this notation we have

$$d \otimes^r (d_1 \vee d_2) = d \otimes^r d_1 \ \vee d \ \otimes^r d_2$$

but in general

$$(d_1 \vee d_2) \otimes^r d \neq d_1 \otimes^r d \ \vee \ d_2 \otimes^r d.$$

3 The Language

Let $\mathcal{N}$ be a set of *channel names*, ranged over by $n,m,\ldots$, and $\mathcal{X}$ a set of *process variables*, ranged over by $X,Y,\ldots$. Then the syntax of the language is given by

Processes	T	::=	$NIL \mid T+T \mid n?F \mid n!C \mid X$
			$\mid FT \mid G(\underline{T}),\ G \in Aux$
Abstractions	F	::=	$(X)T$
Concretions	C	::=	$[T]T$

where Aux is a set of auxiliary operators. In this paper we use a particular set of such operators which are defined as follows:

1. *parallelism*
   for each pair of subsets, $\mathcal{A}, \mathcal{B}$ of $\mathcal{N}$, a binary infix parallel operator $_{\mathcal{A}}|_{\mathcal{B}}$

2. *renaming*
   for each function r from $\mathcal{N}$ to $\mathcal{N}$ which is almost everywhere the identity a unary postfix renaming operator $\{r\}$

In $(X)T$ the prefix (X) acts as a binder for occurrences of X in T and this leads to the standard notion of free and bound occurrences of variables, α-conversion and of substitution: $T\{U/X\}$ stands for the term obtained by substituting all free occurrences of X in T by U where as usual the bound variables in T are renamed via α-conversion if necessary so that no free variables in U are captured. More generally if ρ is a substitution, i.e. a mapping from $\mathcal{X}$ to terms of type process, $T\rho$ denotes the result of replacing all free occurrences of each X in T by $\rho(X)$. We use *process* to mean a closed process–term from this language and $P, Q, \ldots$ are used to denote typical processes.

The language may be considered as an extension of *CHOCS*, [Tho90]. The CHOCS processes $a?X.P, a!P.Q$ are represented here by $a?(X)P$, $a![P]Q$ respectively, the parallel *CHOCS* term $P \mid Q$ by $P\,_{\mathcal{N}}|_{\mathcal{N}}\,Q$ and the restriction $P\backslash a$ by $P\,_{\mathcal{A}}|_{\mathcal{N}}\,NIL$ where $\mathcal{A} = \mathcal{N} - \{a\}$. So informally we shall view CHOCS via this representation as a sublanguage.

The operational semantics of the language is given in Figure 1 where for convenience we have omitted the symmetric counterparts to the Choice and Parallelism rules and the function *name* used in the latter has the obvious definition. There are three types of judgements, of the form

$$P \xrightarrow{n?} F$$
$$P \xrightarrow{n!} C$$
$$P \xrightarrow{\tau} Q,$$

where P and Q are processes, F is a closed abstraction term and C a closed concretion term. The relations $\xrightarrow{n?}$ and $\xrightarrow{n!}$ describe the *communication capabilities* of processes while $\xrightarrow{\tau}$ describes the affect of an actual communication; $P \xrightarrow{\tau} Q$ means that P may perform a communication and thereby be transformed into Q.

The crucial rule in Figure 1 is that of communication. As an example of its application we have

$$n?F\ _{\mathcal{A}}|_{\mathcal{B}}\ n![P]Q \xrightarrow{\tau} F\,P\ _{\mathcal{A}}|_{\mathcal{B}}\ Q.$$

This follows because $n![P]Q$ has the capability $n![P]Q \xrightarrow{n!} [P]Q$, i.e. it can send a communication to the channel n and if asked to do so will send the process P and its residual will be Q, while $n?F$ can receive a communication from the channel n and if asked to do so will apply the abstraction F to the incoming value. The net effect of the Communication rule is that in the process $P\ _{\mathcal{A}}|_{\mathcal{B}}\ Q$ the sub-processes P and Q can communicate using any channel they have in common while the Parallelism rule says that it has all the capabilities of P which use channels from $\mathcal{A}$ and all those of Q which use the channels from $\mathcal{B}$. The behaviour of the remaining operators are well-known, either from *CCS*, [Mil89], or in the case of Application from the λ-calculus.

In our language the local scoping of channel names is carried out by the parallel operator $_{\mathcal{A}}|_{\mathcal{B}}$. In $P\ _{\mathcal{A}}|_{\mathcal{B}}\ Q$ all occurrences of names not from $\mathcal{A}$ in P are local to P and

Input	$n?F \xrightarrow{n?} F$	
Output	$n!C \xrightarrow{n!} C$	
Choice	$P \xrightarrow{c} A$	implies $P + Q \xrightarrow{c} A$
Application	$T\{Q/X\} \xrightarrow{c} A$	implies $FQ \xrightarrow{c} A$ where F is $(X)T$
Parallelism	$P \xrightarrow{c} A, \quad name(c) \in \mathcal{A} \cup \{\tau\}$	implies $P \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} Q \xrightarrow{c} A \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} Q$
Communication	$P \xrightarrow{n?} F$ $Q \xrightarrow{n!} [Q_1]Q_2$	implies $P \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} Q \xrightarrow{\tau} FQ_1 \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} Q_2$
Renaming	$P \xrightarrow{c} A$	implies $P\{r\} \xrightarrow{r(c)} A\{r\}$

where

if F is $(X)T$ then $F \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} P$ denotes $(X)(T \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} P)$ (and similarly for $P \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} F$) and $F\{r\}$ denotes $(X)(T\{r\})$

if C is $[Q_1]Q_2$ then $C \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} P$ is $[Q_1](Q_2 \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} P)$ (and similarly for $P \,{}_{\mathcal{A}}\!\mid_{\mathcal{B}} C$) and $C\{r\}$ denotes $[Q_1](Q_2\{r\})$

Figure 1: The Operational semantics

similarly for $\mathcal{B}$ in Q. With this idea of channel scoping it turns out that our operational semantics implements a dynamic scoping mechanism. To see this let us use the more usual notation $P \mid Q$ for $P \,{}_{\mathcal{N}}\!\mid_{\mathcal{N}} Q$ and $P\backslash\mathcal{A}$ in place of $P \,{}_{\mathcal{B}}\!\mid_{\mathcal{N}} NIL$ where $\mathcal{B}$ is $\mathcal{N} - \mathcal{A}$; this latter process acts like P except that all channels in $\mathcal{A}$ are local. Now consider the term

$$n?(X)(X \mid P) \;\mid\; (n![Q]R)\backslash\mathcal{A}$$

where Q is a process using some channels from $\mathcal{A}$. The occurrences of channels from $\mathcal{A}$ in Q are governed by the restriction, i.e. they are local. Now a communication using the channel n is possible and when it happens the process is transformed into

$$(X)(X \mid P)Q \;\mid\; (R)\backslash\mathcal{A}.$$

The result is that Q has escaped from the restriction by being sent from one process to the other. As another example consider

$$n?(X)((X \mid P)\backslash\mathcal{A}) \;\mid\; n![Q]R.$$

Here Q is not governed by the restriction but the effect of the communication is to transform the process into

$$(X)((X \mid P)\backslash\mathcal{A})Q \mid R$$

Because of the operational semantics of function application this has exactly the same behaviour as

$$(Q \mid P)\backslash\mathcal{A} \mid R$$

where now all occurrences of channels from $\mathcal{A}$ in Q are considered local.

Based on this operational semantics we give three different behavioural equivalences or preorders. The first is motivated from the view of the lazy λ-calculus advocated in [Abr90, Bou90b]. Let $\stackrel{\varepsilon}{\Longrightarrow}$ be the reflexive transitive closure of the relation $\stackrel{\tau}{\longrightarrow}$ and for c of the form $n?$, $n!$ or τ let $\stackrel{c}{\Longrightarrow}$ denote $\stackrel{\varepsilon}{\Longrightarrow} \circ \stackrel{c}{\longrightarrow}$. We write $P \stackrel{c}{\Longrightarrow}$ to mean that there exists some A such that $P \stackrel{c}{\Longrightarrow} A$. This formalises a basic observation one can make of a process, namely its capability of performing a communication. A direct comparison of capabilities is given by

$$P \prec Q \text{ if for every } c \text{ of the form } n! \text{ or } n?\ P \stackrel{c}{\Longrightarrow} \text{ implies } Q \stackrel{c}{\Longrightarrow}$$

A behavioural preorder may now be defined by comparing capabilities in all possible contexts. A context $C[\ ]$ is simply any term which contains one occurrence of a "hole" $[\ ]$.

Definition 3.1 $P \sqsubseteq_{\mathcal{O}} Q$ if $C[P] \prec C[Q]$ for every closed context $C[\ ]$. □

This is a generalisation of the *observation* preorder defined for the λ-calculus in [Bou90b, Abr90] although for simplicity we confine our attention to closed terms. There, as explained in the introduction, the basic observation one can make of a λ-term p is its ability to *converge*, i.e. reduce to a functional term of the form $\lambda x \ldots$.; this is denoted by $p \Downarrow$. Then the observation preorder between λ-terms is defined as above: $p \sqsubseteq_{\mathcal{O}} q$ if for every closing context $C[\ ]$ $C[p] \prec C[q]$ where as before $\prec$ is defined in terms of the basic observation: $p \prec q$ if $p \Downarrow$ implies $q \Downarrow$.

The generalisation is even more direct if we define the basic observation to be the ability to perform *some* action: Let $P \Longrightarrow$ mean that there is some n such that $P \stackrel{n!}{\Longrightarrow}$ and let $P \prec' Q$ be defined as $P \Longrightarrow$ implies $Q \Longrightarrow$. Then let $P \sqsubseteq'_{\mathcal{O}} Q$ if for every closed context $C[\ ]$ $C[P] \prec' C[Q]$. However the two resulting behavioural preorders are equivalent.

Proposition 3.2 *$P \sqsubseteq_{\mathcal{O}} Q$ if and only if $P \sqsubseteq'_{\mathcal{O}} Q$.*

Proof: The "only if" direction is obvious. So suppose $P \sqsubseteq'_{\mathcal{O}} Q$ and $P \stackrel{c}{\Longrightarrow}$ for some c. As an example take the case when c is $n!$. Then consider the context $C[\ ] = [\ ]_{\emptyset|\{m\}}\, n?(X)m![NIL]NIL$ where m is any channel different from n. Then $C[P] \Longrightarrow$ and since $P \sqsubseteq'_{\mathcal{O}} Q$ it follows that $C[Q] \Longrightarrow$. But this is only possible if $Q \stackrel{n!}{\Longrightarrow}$. □

The second behavioural preorder is a direct application of the general framework of *testing*, [Hen88], originally developed for process algebras. Here the idea is that two processes are deemed to be equivalent unless there is a test or experiment which

distinguishes between them. Processes are considered to be independent entities or "black boxes" and a test consists of a series of interactions between the process and the tester which continue until such time as the the tester reaches what it considers to be a successful state. The tester has no control over the process; it simply tries to communicate with the process, the process may deem to reply and if it does the tester may proceed with the experiment in a manner dependent on the reply of the process. In the present setting we take as a test any process which may use a new distinguished name ω and say it is in a successful state if it can perform the action $\omega!$. The application of the test E to the process P is defined to be a maximal sequence of the form

$$E \mid P = E_0 \mid P_0 \stackrel{\tau}{\longrightarrow} E_1 \mid P_1 \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} E_k \mid P_k \stackrel{\tau}{\longrightarrow} \ldots$$

and it is a successful application if there exists an $n \geq 0$ such that $E_n \stackrel{\omega!}{\longrightarrow}$. Then we say P *may* E if there exists an application of E to P which is successful. Finally

Definition 3.3 $P \sqsubseteq_{\mathcal{T}} Q$ if for every test E P *may* E implies Q *may* E. □

This preorder has an alternative characterisation which will later prove useful.

Proposition 3.4 *$P \sqsubseteq_{\mathcal{T}} Q$ if and only if for every n not appearing in P, Q and for every process R, $R \,{}_{\{n\}}|_{\emptyset}\, P \stackrel{n!}{\Longrightarrow}$ implies $R \,{}_{\{n\}}|_{\emptyset}\, Q \stackrel{n!}{\Longrightarrow}$*

Proof: Note that P *may* E if and only if $E \,{}_{\{\omega\}}|_{\emptyset}\, P \stackrel{w!}{\Longrightarrow}$ since it is assumed that ω does not occur in P. One direction follows immediately. The other direction is a consequence of the fact that $R \,{}_{\{n\}}|_{\emptyset}\, P \stackrel{n!}{\Longrightarrow}$ if and only if $(n?(X)w![NIL]NIL \,{}_{\{\omega\}}|_{N}\, R) \;{}_{\{\omega\}}|_{\emptyset}\; P \stackrel{w!}{\Longrightarrow}$ provided n does not occur in P. □

Applying a test to a process may be interpreted as observing its capability to perform $\omega!$ in a restricted context. For this reason we have

Proposition 3.5 *$P \sqsubseteq_{\mathcal{C}} Q$ implies $P \sqsubseteq_{\mathcal{T}} Q$.*

Proof: This follows from the fact that P *may* E if and only if $C[P] \stackrel{\omega!}{\Longrightarrow}$ where $C[\,]$ is the context $E \,{}_{\{\omega\}}|_{\emptyset}\, [\,]$. □

Observing the ability of a process to perform an action in a given context may also be viewed as a form of test but in this case the tester, namely the context, has considerable power over the process being tested. For example a test of the form $n![[\,]]P \mid Q$ has the ability to send the process being tested to the subprocess Q to be examined. Q may have the form $n?(X) \ldots X \ldots X \ldots$ and so when it receives the process copies of it may be in turn be distributed to subprocesses for examination. So apriori the observation preorder $\sqsubseteq_{\mathcal{C}}$ is much finer than the testing preorder $\sqsubseteq_{\mathcal{T}}$. Nevertheless we show that they coincide. This result may be viewed as a generalisation of the *Context Lemma* for the λ-calculus, see Proposition 2.3.6 of [AO89].

The third behavioural preorder is based on a "realizability" interpretation for the language using a generalised modal language $\mathcal{L}$. Here the motivation comes from two

different areas of research. On the one hand there is the well-known modal characterisation of the behavioural equivalence called *bisimulation equivalence* using the modal language *HML*, [Mil89]. On the other there are the interpretations of *property-logics* in terms of the λ-calculus in work such as [Abr90, Bou91, Hin83, BCDC83].
For each of the syntactic categories, processes, abstractions and concretions, we have a corresponding set of formulae, $\mathcal{L}^P$, $\mathcal{L}^F$, $\mathcal{L}^C$ respectively. These are defined by

$$\begin{array}{lll}
\text{Processes :} & \omega \in \mathcal{L}^P & \\
& \phi \in \mathcal{L}^F & \text{implies } \langle n? \rangle \phi \in \mathcal{L}^P \\
& \phi \in \mathcal{L}^C & \text{implies } \langle n! \rangle \phi \in \mathcal{L}^P \\
\text{Abstractions :} & \phi_1, \ldots, \phi_k, \psi \in \mathcal{L}^P & \text{implies } \phi_1, \ldots, \phi_k \rightarrow \psi \in \mathcal{L}^F \\
\text{Concretions :} & \phi_1, \ldots \phi_k, \psi \in \mathcal{L}^P & \text{implies } [\phi_1, \ldots \phi_k]\psi \in \mathcal{L}^C
\end{array}$$

For convenience we introduce the obvious vector notation for formulae so that for example the formation rule for concretion formulae may be shortened to

$$\underline{\phi}, \psi \in \mathcal{L}^P \text{ implies } [\underline{\phi}]\psi \in \mathcal{L}^C .$$

The satisfaction relation, $\models^{\mathcal{O}}$ is also typed in that it only defined between closed terms of each syntactic category and modal formulae from the corresponding language. It is extended in the natural way to vectors of formulae so that for example $P \models^{\mathcal{O}} \underline{\phi}$ means that for each i, $1 \leq i \leq k$, $P \models^p \phi_i$. It is defined by induction over formulae as follows:

$P \models^{\mathcal{O}} \omega$ for every process P

$P \stackrel{n?}{\Longrightarrow} F$, $F \models^{\mathcal{O}} \phi$ implies $P \models^{\mathcal{O}} \langle n? \rangle \phi$

$P \stackrel{n!}{\Longrightarrow} C$, $C \models^{\mathcal{O}} \phi$ implies $P \models^{\mathcal{O}} \langle n! \rangle \phi$

if $(\forall P,\ P \models^{\mathcal{O}} \underline{\phi}$ implies $FP \models^{\mathcal{O}} \psi)$ then $F \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$

if $Q_1 \models^{\mathcal{O}} \underline{\phi}$ and $Q_2 \models^{\mathcal{O}} \psi$ then $[Q_1]Q_2 \models^{\mathcal{O}} [\underline{\phi}]\psi$

Note that $\models^{\mathcal{O}}$ depends entirely on the behaviour of terms and not on their structure.

Let $\mathcal{L}(P) = \{ \phi \mid P \models^{\mathcal{O}} \phi \}$ i.e. $\mathcal{L}(P)$ is the set of all formulae satisfied by the process P.

Definition 3.6 Let $P \ \sqsubseteq_{\mathcal{L}} Q$ if $\mathcal{L}(P) \subseteq \mathcal{L}(Q)$. □

We aim to show that the three behavioural preorders, $\sqsubseteq_{\mathcal{O}}$, $\sqsubseteq_{\mathcal{T}}$ and $\sqsubseteq_{\mathcal{L}}$ coincide. The modal language is carefully chosen with this in mind and apparently simple changes would make the result untrue. For example we do not have $\wedge$ as a connective and adding it would change the relation $\sqsubseteq_{\mathcal{L}}$. As an example consider the processes P_1, Q_1 defined respectively by $m![NIL](n! + k!)$ and $m![NIL]n! + m![NIL]k!$ where we use $m!$ as a shorthand for the process $m![NIL]NIL$, whenever it makes sense. It turns out that $P_1 \ \sqsubseteq_{\mathcal{O}} Q_1$ but $P_1 \models^{\mathcal{O}} \phi$ and $Q_1 \not\models^{\mathcal{O}} \phi$ where ϕ is $\langle m! \rangle [\omega](\langle n! \rangle [\omega]\omega \wedge \langle k! \rangle [\omega]\omega)$. Similarly

sequences are necessary in the constructions $[\underline{\phi}]\psi$ and $\underline{\phi} \rightarrow \psi$. For consider P_2, Q_2 defined by $m![\ n! + k!\ ]NIL$ and $m![\ n!\ ]NIL + m![\ k!\ ]NIL$ respectively. Then $P_2 \models^{\mathcal{O}} \phi$ if and only if $Q_2 \models^{\mathcal{O}} \phi$ for every ϕ not using sequences but $P_2 \not\simeq_{\mathcal{L}} Q_2$.

The modal language is in fact determined by a denotational model which provides a crucial link in establishing the equality between the behavioural preorders. The model and the denotational interpretation of the language in it is described in the next two sections. We then show that this model is *fully abstract* with respect to the three behavioural preorders.

4 The Model

Consider the domain equation

$$\begin{array}{lcll} \mathbf{P} & = & (\mathcal{N} \longrightarrow \mathbf{C}_\perp) \times (\mathcal{N} \longrightarrow \mathbf{F}_\perp) & \text{Processes} \\ \mathbf{F} & = & [\mathbf{P} \longrightarrow \mathbf{P}] & \text{Abstractions} \\ \mathbf{C} & = & \mathbf{P} \otimes^r \mathbf{P} & \text{Concretions} \end{array}$$

Intuitively this models a process using two functions one, from $\mathcal{N}$ to $\mathbf{C}_\perp$, representing its output potential on each of the possible channels and the other, from $\mathcal{N}$ to $\mathbf{F}_\perp$, representing its input potential. Of course a process has a set of potentials on each channel but this "set-theoretical union" of potentials will be modelled using the join operation $\bigvee$. It may also be that for a particular name a process may have no associated input or output potential. For this reason we use $\mathbf{C}_\perp$ and $\mathbf{F}_\perp$ rather than $\mathbf{C}$ and $\mathbf{F}$ as the corresponding functions can map such names to $\perp$; recall that the set of total functions from $\mathcal{N}$ to $D_\perp$ is isomorphic to the set of partial functions from $\mathcal{N}$ to D. Terms of type abstraction are interpreted as functions from processes to processes while concretions are interpreted as tensor products of processes. It is tempting to use the simpler Cartesian product but this would result in the identification of the two processes P_2, Q_2, defined in the previous section, which are behaviourally different.

This equation has an initial solution in the standard category of *cpos* with embeddings, [Plo81], and since the constructions preserve domains the solution is also a domain. We will omit all references to the isomorphisms generated by this construction. Thus for example we will consider an element p of $\mathbf{P}$ as a pair of functions which we denote by $p_!$ and $p_?$ respectively. Moreover for each n in $\mathcal{N}$ it will be convenient to denote $p_!(n)$ and $p_?(n)$ as $p(n!)$ and $p(n?)$ respectively. To illustrate the convenience of this informal view of these domains we define two functions which will be useful in later developments.

For each $n \in \mathcal{N}$ let the function $n_{out}\colon \mathbf{C} \longmapsto \mathbf{P}$ be defined by

$$n_{out}(c) = < \lambda x \in \mathcal{N}.x = n \rightarrow c_\perp, \perp\ ,\ \lambda x \in \mathcal{N}.\perp >$$

and let $n_{in}\colon \mathbf{F} \longmapsto \mathbf{P}$ be defined in an analogous manner:

$$n_{in}(f) = < \lambda x \in \mathcal{N}.\perp, \lambda x \in \mathcal{N}.x = n \rightarrow f_\perp, \perp >$$

Proposition 4.1 *The functions n_{out}, n_{in} are linear.*

Proof: By calculation. □

These domains are completely determined by their primes which we now proceed to describe. For $\mathbf{A} = \mathbf{P}, \mathbf{F}, \mathbf{C}$ respectively, let $\mathbf{A}_{\mathcal{KP}}$ be the least subsets of $\mathbf{A}$ satisfying

1. $\perp \in \mathbf{A}_{\mathcal{KP}}$
2. $c \in \mathbf{C}_{\mathcal{KP}}$ implies $n_{out}(c) \in \mathbf{P}_{\mathcal{KP}}$ for all $n \in \mathcal{N}$
3. $f \in \mathbf{F}_{\mathcal{KP}}$ implies $n_{in}(f) \in \mathbf{P}_{\mathcal{KP}}$ for all $n \in \mathcal{N}$
4. $d_1, \ldots, d_k, e \in \mathbf{P}_{\mathcal{KP}}$ implies $d_1 \vee \ldots \vee d_k \Rightarrow e \in \mathbf{F}_{\mathcal{KP}}$
5. $d_1, \ldots, d_k, e \in \mathbf{P}_{\mathcal{KP}}$ implies $(d_1 \vee \ldots \vee d_k) \otimes^r e \in \mathbf{C}_{\mathcal{KP}}$.

Theorem 4.2 *For* $\mathbf{A} = \mathbf{P}, \mathbf{F}, \mathbf{C}$ *respectively,* $\mathcal{KP}(\mathbf{A}) = \mathbf{A}_{\mathcal{KP}}$.

Proof: (Outline) It is very easy to prove by induction that if $d \in \mathbf{A}_{\mathcal{KP}}$ then d is prime. To prove the converse we must calculate the primes of the domains $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{F}$. Let $\mathcal{F}$ be the domain constructor for the domains. This can be represented as a triple of functors

$$\begin{aligned} \mathcal{F}_{\mathbf{P}}(P,F,C) &= (\mathcal{N} \longrightarrow C_\perp) \times (\mathcal{N} \longrightarrow F_\perp) \\ \mathcal{F}_{\mathbf{F}}(P,F,C) &= [P \longrightarrow P] \\ \mathcal{F}_{\mathbf{C}}(P,F,C) &= P \otimes^r P \end{aligned}$$

Then the domains ($\mathbf{P}$, $\mathbf{F}$, $\mathbf{C}$) are constructed as the inverse limit of the sequence of domains

$$\begin{aligned} (\mathbf{P}^0, \mathbf{F}^0, \mathbf{C}^0) &= (\perp, \perp, \perp) \\ (\mathbf{P}^{k+1}, \mathbf{F}^{k+1}, \mathbf{C}^{k+1}) &= \mathcal{F}(\mathbf{P}^k, \mathbf{F}^k, \mathbf{C}^k) \end{aligned}$$

where here $\perp$ denotes the trivial domain consisting of one element. Moreover the prime elements of the domains $\mathbf{A}$ are the (embeddings) of the prime elements of the "approximations" $\mathbf{A}^k$.
So we can prove by induction on k that $p \in \mathbf{A}^k$ implies $p \in \mathbf{A}_{\mathcal{KP}}$. □

An element of the domain $\mathbf{A}$ is determined by the set of primes it dominates and following [Abr90] we could view the primes as properties or propositions. Such a proposition is "satisfied" by an element of $\mathbf{A}$ if the element dominates it and in this way elements are determined by the propositions they satisfy. If we invent a logical language for describing these propositions we then have a logical presentation of the domain, [Sco82].

The logical language we choose is $\mathcal{L}$ introduced in the previous section. We could devise a proof system for this language with judgements of the form

$$\phi \leq \psi$$

General :

Refl	$\phi \leq \phi$
Weak	$\dfrac{\underline{\phi} \leq \psi}{\underline{\phi},\ \phi' \leq \psi}$
Trans	$\dfrac{\underline{\phi} \leq \underline{\psi},\ \ \underline{\psi} \leq \xi}{\underline{\phi} \leq \xi}$

Processes :

$\mathcal{L}P_1$	$\phi \leq \omega$
$\mathcal{L}P_2$	$\dfrac{\phi \leq \psi}{\langle c \rangle \phi \ \leq \ \langle c \rangle \psi}$

Abstractions :

$\mathcal{L}F_1$	$\phi \leq (\omega \rightarrow \omega)$
$\mathcal{L}F_2$	$\dfrac{\underline{\phi} \leq \underline{\phi'},\ \ \psi \leq \psi'}{\underline{\phi'} \rightarrow \psi \leq \underline{\phi} \rightarrow \psi'}$

Concretions :

$\mathcal{L}C_1$	$\dfrac{\underline{\phi} \leq \underline{\phi'},\ \ \psi \leq \psi'}{[\underline{\phi}]\psi \leq [\underline{\phi'}]\psi'}$

Figure 2: The proof system for $\mathcal{L}$

but in order to have slightly simpler proof rules we use judgements of the form

$$\phi_1, \ldots, \phi_k \leq \psi$$

where all the formulae are assumed to be of the same type. Using vector notation judgements are abbreviated to $\underline{\phi} \leq \psi$ which may be read as the conjunction of $\underline{\phi}$ implies ψ. The proof system is given in Figure 2 where in the rules Trans, $\mathcal{L}F_2$ and $\mathcal{L}C_1$ we use $\underline{\phi} \leq \underline{\phi'}$ as an abbreviation for the k judgements $\underline{\phi} \leq \phi'_i$, $1 \leq i \leq k$ where k is the length of the vector ϕ'. We write $\mathcal{L} \vdash \underline{\phi} \leq \psi$ if $\underline{\phi} \leq \psi$ can be derived in the proof system and more generally $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$ if $\underline{\phi} \leq \psi_j$ can be derived for each j.

We now show how to interpret the formulae as primes in the appropriate models by

defining a map $[\![\ ]\!]: \mathcal{L}^A \longmapsto \mathcal{KP}(\mathbf{A})$. Then the statement $\mathcal{L} \vdash \underline{\phi} \leq \psi$ may be interpreted semantically as saying that $[\![\psi]\!]$ is dominated by the element $[\![\phi_1]\!] \vee \ldots \vee [\![\phi_k]\!]$ and since $[\![\psi]\!]$ will be a prime this means that there is some i such that $[\![\psi]\!] \leq [\![\phi_i]\!]$. For convenience we use $[\![\underline{\phi}]\!]$ to denote $[\![\phi_1]\!] \vee \ldots \vee [\![\phi_k]\!]$.

Definition 4.3

$$\begin{array}{lll} \text{Processes :} & [\![\omega]\!] & = \ \bot \\ & [\![\langle n! \rangle \phi]\!] & = \ n_{out}([\![\phi]\!]) \\ & [\![\langle n? \rangle \phi]\!] & = \ n_{in}([\![\phi]\!]) \\ \text{Abstractions :} & [\![\underline{\phi} \rightarrow \psi]\!] & = \ [\![\underline{\phi}]\!] \Rightarrow [\![\psi]\!] \\ \text{Concretions :} & [\![[\underline{\phi}]\psi]\!] & = \ [\![\underline{\phi}]\!] \otimes^r [\![\psi]\!] \end{array}$$

□

Using this interpretation

Theorem 4.4 *For* $\mathbf{A} = \mathbf{P}$, $\mathbf{C}$, $\mathbf{F}$ *respectively*

1. *The map* $[\![\ ]\!]: \mathcal{L}^A \longmapsto \mathcal{KP}(\mathbf{A})$ *is surjective, i.e. for every* $p \in \mathcal{KP}(\mathbf{A})$ *there exists a formula* $\phi \in \mathcal{L}^A$ *such that* $[\![\phi]\!] = p$

2. $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$ *if and only if* $[\![\underline{\psi}]\!] \leq [\![\underline{\phi}]\!]$.

Proof: It is straightforward to show by induction that for every $p \in \mathbf{A}_{\mathcal{KP}}$ there exists a formula $\phi \in \mathcal{L}^A$ such that $[\![\phi]\!] = p$. For example if p has the form $n_{in}(f)$ and is in $\mathbf{P}_{\mathcal{KP}}$ because $f \in \mathbf{F}_{\mathcal{KP}}$ then by induction we may assume that the exists a $\psi \in \mathcal{L}^F$ such that $[\![\psi]\!] = f$. It follows that $[\![\langle n? \rangle \psi]\!] = p$.

The proof that $\mathcal{L} \vdash \underline{\phi} \leq \psi$ implies $[\![\psi]\!] \leq [\![\underline{\phi}]\!]$ is equally straightforward. It proceeds by induction on the proof of $\underline{\phi} \leq \psi$ and this immediately implies the corresponding result for vectors, namely if $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$ then $[\![\underline{\psi}]\!] \leq [\![\underline{\phi}]\!]$. We prove the converse and it is sufficient to prove $[\![\psi]\!] \leq [\![\phi]\!]$ implies $\mathcal{L} \vdash \phi \leq \psi$. For suppose we have established this and that $[\![\underline{\psi}]\!] \leq [\![\underline{\phi}]\!]$. This means that $[\![\psi_i]\!] \leq [\![\underline{\phi}]\!]$ for each i and since $[\![\psi_i]\!]$ is prime this implies $[\![\psi_i]\!] \leq [\![\phi_j]\!]$ for some j. Applying the result we obtain $\mathcal{L} \vdash \phi_j \leq \psi_i$ and by the rule weakening $\mathcal{L} \vdash \underline{\phi} \leq \psi_i$. Since this is true for each i it follows by definition that $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$.

The proof that $[\![\psi]\!] \leq [\![\phi]\!]$ implies $\mathcal{L} \vdash \phi \leq \psi$ proceeds by induction on the structure of ψ.

1. $\psi = \omega$
   Use Rule $\mathcal{L}P_1$

2. $\psi = \langle n? \rangle \eta$
   Note that since $[\![\psi]\!] \leq [\![\phi]\!]$ it follows that ϕ must be of the form $\langle n? \rangle \xi$; otherwise $[\![\phi]\!](n?) = \bot$ and so $[\![\phi]\!]$ would not dominate $[\![\psi]\!]$. Moreover it is easy to check that $[\![\eta]\!] \leq [\![\xi]\!]$ and therefore by induction $\mathcal{L} \vdash \xi \leq \eta$. Then using the rule $\mathcal{L}P_2$ we obtain the required $\mathcal{L} \vdash \phi \leq \psi$.
   The case when $\psi = \langle n! \rangle \eta$ is similar.

3. $\psi = [\underline{\psi^1}]\psi^2$
   Let ϕ have the form $[\underline{\phi^1}]\phi^2$. So $[\![\underline{\psi^1}]\!] \otimes^r [\![\psi^2]\!] \leq [\![\underline{\phi^1}]\!] \otimes^r [\![\phi^2]\!]$ and since $i^{\otimes^r}$ is

injective it follows that $[\![\underline{\phi^1}]\!] \leq [\![\underline{\psi^1}]\!]$ and $[\![\phi^2]\!] \leq [\![\psi^2]\!]$. We can apply induction to obtain $\mathcal{L} \vdash \underline{\phi^1} \leq \underline{\psi^1}$ and $\mathcal{L} \vdash \phi^2 \leq \psi^2$ and an application of the rule $\mathcal{L}C_1$ yields $\mathcal{L}[\underline{\phi^1}]\phi^2 \leq [\underline{\psi^1}]\psi^2$.

4. $\psi = \underline{\psi^1} \to \psi^2$
Let ϕ have the form $\underline{\phi^1} \to \phi^2$. So we have $[\![\underline{\psi^1}]\!] \Rightarrow [\![\psi^2]\!] \leq [\![\underline{\phi^1}]\!] \Rightarrow [\![\phi^2]\!]$. There are two cases to consider

   (a) $[\![\psi^2]\!] = \bot$.
   Every formula other than ω has a non-trivial interpretation and therefore ψ^2 must be ω. From the rule $\mathcal{L}F_1$ we have $\underline{\phi^1} \to \phi^2 \leq \omega \to \omega$ while the rules $\mathcal{L}F_2$, $\mathcal{L}P_1$ and Weak give $\omega \to \omega \leq \underline{\psi^1} \to \omega$ from which the required $\mathcal{L} \vdash \underline{\phi^1} \to \phi^2 \leq \underline{\psi^1} \to \omega$ follows.

   (b) $[\![\psi^2]\!] \neq \bot$
   This means that $[\![\underline{\phi^1}]\!] \leq [\![\underline{\psi^1}]\!]$ and $[\![\psi^2]\!] \leq [\![\phi^2]\!]$. An application of induction followed by the rule $\mathcal{L}F_2$ once more gives the required result.

□

This theorem states that the models are in fact determined by the language $\mathcal{L}$ together with it proof system. We now state this precisely. An **A**-*filter* is a non-empty subset F of vectors of formulae from $\mathcal{L}^A$ with the property that $\underline{\phi} \in F$ and $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$ implies $\underline{\psi} \in F$. Let Fil^A be the set of all **A**-filters ordered by set inclusion.

Theorem 4.5 *Fil^A is a domain and isomorphic to* **A**.

Proof: It is easy to check that $\bigvee_I F_i$ is given by $\{\omega_A\} \cup \bigcup_I F_i$ where ω_P, ω_F and ω_C are ω, $\omega \to \omega$ and $\omega[\omega]$ respectively. Moreover the compact elements are all those filters of the form $\{\, \phi \mid \underline{\psi} \leq \phi \,\}$ for some vector $\underline{\psi}$ and the primes all have the form $\{\, \phi \mid \psi \leq \phi \,\}$ for some single formula ψ.

Let $f\colon \mathbf{A} \longmapsto Fil^A$ be defined by $f(x) = \{\, \phi \mid [\![\phi]\!] \leq x \,\}$ and $g\colon Fil^A \longmapsto \mathbf{A}$ by $g(F) = \bigvee\{\, [\![\phi]\!] \mid \phi \in F \,\}$. It follows more or less immediately from the previous theorem that f and g determine an isomorphism. □

We end this section with a soundness result for the logic with respect to the realizability interpretation of the previous section.

Proposition 4.6 *For* $\mathbf{A} = \mathbf{P}$, $\mathbf{F}$, $\mathbf{C}$ *respectively* $\mathcal{L} \vdash \underline{\phi} \leq \psi$ *and* $A \models^{\mathcal{O}} \underline{\phi}$ *implies* $A \models^{\mathcal{O}} \psi$.

Proof: By induction on the length of the proof of $\mathcal{L} \vdash \underline{\phi} \leq \psi$. □

5 The Interpretation of the Language

Using the model of the previous section we may interpret the language in a standard fashion. Let ENV be the set of *environments*, i.e. mappings from $\mathcal{X}$ to $\mathbf{P}$, ranged over by σ. Then for each term T of type $\mathbf{A}$ we define $[\![T]\!]_A$: $ENV \longmapsto \mathbf{A}$ as follows:

- $[\![NIL]\!]_P\sigma = \bot$
- $[\![n?F]\!]_P\sigma = n_{in}([\![F]\!]_F\sigma)$
- $[\![n!C]\!]_P\sigma = n_{out}([\![C]\!]_C\sigma)$
- $[\![X]\!]_P\sigma = \sigma(X)$
- $[\![T + U]\!]_P\sigma = [\![T]\!]_P\sigma \vee [\![U]\!]_P\sigma$
- $[\![FT]\!]_P\sigma = [\![F]\!]_F\sigma([\![T]\!]_P\sigma)$
- $[\![(X)T]\!]_F\sigma = \lambda d \in \mathbf{P}.[\![T]\!]_P\sigma[X \mapsto d]$
- $[\![[T]U]\!]_C\sigma = [\![T]\!]_P\sigma \otimes^r [\![U]\!]_P\sigma$
- $[\![G(\underline{T})]\!]_P\sigma = g([\![\underline{T}]\!]_P)$
  where for each auxiliary function symbol G we have a corresponding function g of the appropriate type.

To complete the interpretation we need to define the functions corresponding the function symbols in Aux. To do so it is convenient to introduce some notational conventions. The first concerns the "lifting" operation. Suppose $t(\underline{x})$ is a meta-expression involving the variables $\underline{x}$ with the property that $t(\underline{v}) \in E$ for all values v_i from a set E^i. Then if $w_i \in E^i_\bot$, $t(\underline{w})$ denotes the value in $E_\bot$ determined by

$$t(\underline{w}) = \begin{cases} \bot & \text{if } \exists i.w_i = \bot \\ t(\underline{v}) & \text{otherwise where } w_i = (v_i)_\bot \end{cases}$$

The second convention is a convenient way of describing functions over tensor products. Let $\lambda(d_1, d_2) \in D_1 \times D_2.t$ represent a right-linear function in $[D_1 \times D_2 \longrightarrow E]$. Then we use $\lambda^{\otimes^r}(d_1, d_2) \in D_1 \times D_2.t$ to represent its unique extension to a linear function in $[D_1 \otimes^r D_2 \longrightarrow E]$.

The most difficult function to define is that corresponding to the parallel operator $_\mathcal{A}|_\mathcal{B}$. Informally the definition simply mimics the usual interleaving interpretation of parallelism. Formally it takes the form $Y\ Par_{\mathcal{A},\mathcal{B}}$ where Y is the least fixpoint operator and $Par_{\mathcal{A},\mathcal{B}}$ is a function of type $[\mathbf{P} \times \mathbf{P} \longrightarrow \mathbf{P}] \longrightarrow [\mathbf{P} \times \mathbf{P} \longrightarrow \mathbf{P}]$. Intuitively if F is of type $\mathbf{P} \times \mathbf{P} \longrightarrow \mathbf{P}$ then $Par_{\mathcal{A},\mathcal{B}}F$, when applied to two processes x and y calculates the resulting process by "unioning" together three different components. The first considers possible moves from x and calculates their residuals by applying F recursively, the second does the same for y while the third calculates the possible results of communication between x and y using any channel in $\mathcal{N}$. Formally $Par_{\mathcal{A},\mathcal{B}}F(x, y)$ is

defined by

$$\begin{array}{ll}
\bigvee_{m\in\mathcal{A}} & m_{in}\lambda d \in \mathbf{P}.F(x(m?)d, y) \\
& \vee\, m_{out}(\ \lambda^{\otimes^r}(d,d') \in D \times D.d \otimes^r F(d',y)\ )x(m!) \\
\vee & \\
\bigvee_{m\in\mathcal{B}} & m_{in}\lambda d \in \mathbf{P}.F(x, y(m?)d) \\
& \vee\, m_{out}(\ \lambda^{\otimes^r}(d,d') \in \mathbf{P} \times \mathbf{P}.d \otimes^r F(x,d')\ )y(m!) \\
\vee & \\
\bigvee_{m\in\mathcal{N}} & (\ \lambda^{\otimes^r}(d,d') \in \mathbf{P} \times \mathbf{P}.F(x(m?)d, d')\ )y(m!) \\
& \vee\, (\ \lambda^{\otimes^r}(d,d') \in \mathbf{P} \times \mathbf{P}.F(d', y(m?)d)\ )x(m!).
\end{array}$$

Renaming is interpreted in a similar manner as $Y\ \ Ren_r$ where for each renaming $r\ \ Ren_r{:}[\mathbf{P} \longrightarrow \mathbf{P}] \longmapsto [\mathbf{P} \longrightarrow \mathbf{P}]$ is defined by

$$\begin{array}{lll}
Ren_r Fx = & \bigvee_{m\in\mathcal{N}} & r(m)_{in}\lambda d \in \mathbf{P}.F(x(m?)d) \\
& & \vee\, r(m!)_{out}(\ \lambda^{\otimes^r}(d,d') \in \mathbf{P} \times \mathbf{P}.d \otimes^r F(d')\ )x(m!).
\end{array}$$

For the remainder of the paper we will omit the subscripts from these semantic functions, rendering $[\![\ ]\!]_A$ simply as $[\![\ ]\!]$. When interpreting closed terms we will also usually omit all reference to environments. We will also make use of the standard substitution lemma for denotational semantics which states $[\![A]\!]\sigma[X \mapsto [\![T]\!]\sigma] = [\![A\{T/X\}]\!]\sigma$.

Proposition 5.1 *For every $G \in$ Aux the corresponding function g is multilinear.*

Proof: As an example we consider the parallel operator. It is sufficient to show that $Par_{\mathcal{A},\mathcal{B}}F$ is multilinear provided F is. Let us introduce some abbreviations by rewriting $Par_{\mathcal{A},\mathcal{B}}F(x,y)$ as

$$\begin{array}{lll}
& \bigvee_{m\in\mathcal{A}} & m_{in}f^m_{in}(x,y) \vee m_{out}f^m_{out}(x,y) \\
\vee & \bigvee_{m\in\mathcal{B}} & m_{in}g^m_{in}(x,y) \vee m_{out}g^m_{out}(x,y) \\
\vee & \bigvee_{m\in\mathcal{N}} & com^m_l(x,y) \vee com^m_r(x,y)
\end{array}$$

Then it is a matter of showing that the functions f^m_i, g^m_i and com^m_i are linear in both x and y. We give two examples:

- f^m_{out} is multilinear.
  $f^m_{out}(x,y)$ may be written as $k^{\otimes^r}_y x(m!)$ where for any z k_z is the function $\lambda(d,d') \in \mathbf{P} \times \mathbf{P}.d \otimes^r F(d',z)$. Since $(x \vee x')(m!) = x(m!) \vee x'(m!)$ and $k^{\otimes^r}_y$ is linear it follows immediately that $f^m_{out}(x \vee x', y) = f^m_{out}(x,y) \vee f^m_{out}(x',y)$.

  Since F is multilinear and $i^{\otimes^r}$ is right–linear it follows that $k_{y\vee y'} = k_y \vee k_{y'}$. A simple consequence of the universal characterisation of the tensor product construction is the fact that for any two functions $(f_1 \vee f_2)^{\otimes^r} = f_1^{\otimes^r} \vee f_2^{\otimes^r}$. It follows that

$k^{\otimes^r}_{y\vee y'} = k^{\otimes^r}_y \vee k^{\otimes^r}_{y'}$. Therefore

$$\begin{aligned} f^m_{out}(x, y \vee y') &= k^{\otimes^r}_{y\vee y'} x(m!) \\ &= k^{\otimes^r}_y x(m!) \quad \vee \quad k^{\otimes^r}_{y'} x(m!) \\ &= f^m_{out}(x, y) \vee f^m_{out}(x, y'). \end{aligned}$$

- com^m_l is multilinear.
  Here let k_z denote the function $\lambda(d, d') \in \mathbf{P} \times \mathbf{P}.F(z(m?)d, d')$. Then $com^m_l(x, y) = k^{\otimes^r}_x y(m!)$ and linearity with respect to y follows immediately from the linearity of $k^{\otimes^r}_x$. As in the previous case the multilinearity of F implies $k_{x\vee x'} = k_x \vee k_{x'}$ and therefore that $k^{\otimes^r}_{x\vee x'} = k^{\otimes^r}_x \vee k^{\otimes^r}_{x'}$. So

$$\begin{aligned} com^m_l(x \vee x', y) &= k^{\otimes^r}_{x\vee x'} y(m!) \\ & \quad k^{\otimes^r}_x y(m!) \vee k^{\otimes^r}_{x'} y(m!) \\ & \quad com^m_l(x, y) \vee com^m_l(x', y) \end{aligned}$$

□

The fact that these semantic functions are multilinear means that they can be characterised by their effect on primes. This gives a simple method of establishing their properties. For example on can show that $x \;{}_{\mathcal{A}}|_{\mathcal{B}}\; y$ is the same as $y \;{}_{\mathcal{B}}|_{\mathcal{A}}\; x$ by examining the case when x and y are prime. As we have seen the primes in the models can be described by formulae from $\mathcal{L}$ and therefore we can also describe these functions entirely in terms of the logic. We use exactly the same symbol to denote the operator in Aux, the corresponding semantic function over the domain and now the corresponding characterising function over formulae. This may sound confusing but the context will always disambiguate its use.

Let $\mathcal{P}_f(\mathcal{L}^p)$ denote the set of finite non-empty subsets of formulae of type Process. Each $s \in \mathcal{P}_f(\mathcal{L}^p)$ can be interpreted as an element of $\mathbf{P}$ in the obvious way:

$$[\![s]\!] = \bigvee \{\, \phi \mid [\![\phi]\!] \in s \,\}.$$

So for each G in Aux of arity k we define a function $G\colon (\mathcal{L}^p)^k \longmapsto \mathcal{P}_f(\mathcal{L}^p)$ with the property that $G([\![\phi_1]\!], \ldots, [\![\phi_k]\!]) = [\![G(\underline{\phi})]\!]$; on the left hand side G represents a semantic function applied to the elements $[\![\phi_1]\!], \ldots, [\![\phi_k]\!]$ while on the right hand side it is a function over formulae.

To make these definitions more compact let us introduce the notation $\langle c\rangle(\underline{\phi})\psi$ to mean $\langle n?\rangle\underline{\phi} \Rightarrow \psi$ when c is $m?$ and $\langle m!\rangle[\underline{\phi}]\psi$ when it is $m!$. Then the function $\{r\}$ over formulae is defined by the two clauses

- $\omega\{r\} = \omega$
- $(\langle c\rangle(\underline{\phi})\psi)\{r\} = r(c)(\underline{\phi})\xi$ where $\xi = \psi\{r\}$.

The main difficulty is in defining the parallel function ${}_{\mathcal{A}}|_{\mathcal{B}}$. This is given in tabular form in Figure 3, where some of the symmetric cases have been omitted.

Theorem 5.2 *For every G in Aux, $[\![G(\underline{\phi})]\!] = G([\![\phi_1]\!], \ldots, [\![\phi_k]\!])$*

α	β	$\alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta$
ω	ω	ω
ω	$\langle c\rangle(\phi)\psi$	$\{\omega\}$ $\cup \{ \langle c\rangle(\phi)\xi \mid \xi \in \psi,\ c \in \mathcal{B} \}$
$\langle n?\rangle\underline{\phi} \to \psi$	$\langle m?\rangle\underline{\phi'} \to \psi'$	$\{\omega\}$ $\cup \{ \langle n?\rangle\underline{\phi} \to \xi \mid \xi \in \psi \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta,\ n \in \mathcal{A} \}$ $\cup \{ \langle n?\rangle\underline{\omega} \to \xi \mid \xi \in \omega \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta,\ n \in \mathcal{A} \}$ $\cup \{ \langle m?\rangle\underline{\phi'} \to \xi \mid \xi \in \alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \psi',\ m \in \mathcal{B} \}$ $\cup \{ \langle m?\rangle\underline{\omega} \to \xi \mid \xi \in \alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \omega,\ m \in \mathcal{B} \}$
$\langle n!\rangle[\underline{\phi}]\psi$	$\langle m!\rangle[\underline{\phi'}]\psi'$	$\{\omega\}$ $\cup\{ \langle n!\rangle[\underline{\phi}]\xi \mid \xi \in \psi \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta,\ n \in \mathcal{A} \}$ $\cup\{ \langle m!\rangle[\underline{\phi'}]\xi \mid \xi \in \alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \psi',\ m \in \mathcal{B} \}$
$\langle n?\rangle\underline{\phi} \to \psi$	$\langle m!\rangle[\underline{\phi'}]\psi'$	$\{\omega\}$ $\cup \{ \langle n?\rangle\underline{\phi} \to \xi \mid \xi \in \psi \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta,\ n \in \mathcal{A} \}$ $\cup \{ \langle n?\rangle\underline{\omega} \to \xi \mid \xi \in \omega \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta,\ n \in \mathcal{A} \}$ $\cup \{ \langle m!\rangle[\underline{\phi'}]\xi \mid \xi \in \alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \psi',\ m \in \mathcal{B} \}$ $\cup \{ \xi \mid \xi \in \omega \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \psi',\ \ m = n \}$ $\cup \{ \xi \mid \xi \in \psi \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \psi',\ \mathcal{L} \vdash \underline{\phi'} \leq \underline{\phi},\ m = n \}$

Figure 3: The Parallel Operator on Formulae

Proof: For each G the proof proceeds by structural induction on $\underline{\phi}$. As an example we consider one case for the parallel operator: we show $[\![\alpha \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ \beta]\!] = [\![\alpha \ {}_{\mathcal{A}}|_{\mathcal{B}} \ \beta]\!]$ when $\alpha,\ \beta$ are $\langle n?\rangle\underline{\phi} \to \psi,\ \langle m!\rangle[\underline{\phi'}]\psi'$ respectively in the case when n is in $\mathcal{A}$, m is in $\mathcal{B}$ and $m = n$.

As in the proof of Theorem 5.1 we may introduce some notation by writing $x \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ y$ as

$$\begin{array}{ll} \bigvee_{k\in\mathcal{A}} & k_{in}f^k_{in}(x,y) \vee k_{out}f^k_{out}(x,y) \\ \bigvee_{k\in\mathcal{B}} & k_{in}g^k_{in}(x,y) \vee k_{out}g^k_{out}(x,y) \\ \bigvee_{k\in\mathcal{N}} & com^k_l(x,y) \vee com^k_r(x,y). \end{array}$$

In this case for each k $[\![\alpha]\!](k!) = [\![\beta]\!](k?) = \bot$. This means in turn that $f^k_{out}([\![\alpha]\!],[\![\beta]\!]) = g^k_{in}([\![\alpha]\!],[\![\beta]\!]) = \ com^k_r([\![\alpha]\!],[\![\beta]\!]) = \bot$ and that for every k different from n $\ com^k_l([\![\alpha]\!],[\![\beta]\!]) = \bot$. Therefore $[\![\alpha]\!] \ {}_{\mathcal{A}}\|_{\mathcal{B}} \ [\![\beta]\!]$ can be simplified to

$$n_{in}f^n_{in}([\![\alpha]\!],[\![\beta]\!]) \vee m_{out}g^n_{out}([\![\alpha]\!],[\![\beta]\!]) \vee \ comm^m_l([\![\alpha]\!],[\![\beta]\!]).$$

Let us also rewrite $[\![\alpha \ {}_{\mathcal{A}}|_{\mathcal{B}} \beta]\!]$ to a convenient form. Because of the linearity of the prefixing functions $n_{in},\ m_{out}$ it may be written as

$$n_{in}[\![S_{11}]\!] \vee n_{in}[\![S_{12}]\!] \vee m_{out}[\![S_2]\!] \vee [\![S_3]\!] \vee [\![S_4]\!]$$

where

$$\begin{array}{rcl} S_{11} & = & \{\, \underline{\phi} \to \xi \mid \xi \in \psi\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \beta \,\} \\ S_{12} & = & \{\, \omega \to \xi \mid \xi \in \omega\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \beta \,\} \\ S_2 & = & \{\, [\underline{\phi}]\xi \mid \xi \in \alpha\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi' \,\} \\ S_3 & = & \{\, \xi \mid \xi \in \psi\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi',\ \mathcal{L} \models \underline{\phi}' \leq \underline{\phi} \,\} \\ S_4 & = & \{\, \xi \mid \xi \in \omega\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi' \,\} \end{array}$$

To prove the result it is therefore sufficient to establish

$$\begin{array}{rcl} f^n_{in}([\![\alpha]\!],[\![\beta]\!]) & = & [\![S_{11}]\!] \vee [\![S_{12}]\!] \\ g^m_{out}([\![\alpha]\!],[\![\beta]\!]) & = & [\![S_2]\!] \\ com^m_l([\![\alpha]\!],[\![\beta]\!]) & = & [\![S_3]\!] \vee [\![S_4]\!]. \end{array}$$

- $f^n_{in}([\![\alpha]\!],[\![\beta]\!])$ is the function $\lambda d \in \mathbf{P}.([\![\underline{\phi}]\!] \Rightarrow [\![\psi]\!]d)\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\beta]\!]$. But a simple analysis shows that this may be written as

$$[\![\underline{\phi}]\!] \Rightarrow ([\![\psi]\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\beta]\!]) \ \vee\ [\![\omega]\!] \Rightarrow ([\![\omega]\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\beta]\!]).$$

  At this point we can use induction. For example

$$\begin{array}{rcll} [\![\underline{\phi}]\!] \Rightarrow ([\![\psi]\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\beta]\!]) & = & [\![\underline{\phi}]\!] \Rightarrow [\![\psi\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \beta]\!]) & \text{by induction} \\ & = & \vee\ \{\, [\![\underline{\phi}]\!] \Rightarrow [\![\xi]\!] \mid \xi \in \psi\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \beta \,\} & \text{since } \Rightarrow \text{ is right-linear} \\ & = & [\![S_{11}]\!]. & \end{array}$$

  In a similar fashion one can show $[\![\omega]\!] \Rightarrow ([\![\omega]\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\beta]\!]) = [\![S_{12}]\!]$.

- Let k_x denote the function $\lambda(d,d') \in \mathbf{P} \times \mathbf{P}.d \otimes^r (x\ {}_{\mathcal{A}}|_{\mathcal{B}}\ d')$. Then

$$\begin{array}{rcll} g^m_{out}([\![\alpha]\!],[\![\beta]\!]) & = & k^{\otimes^r}_{[\![\alpha]\!]}([\![\beta]\!](m!)) & \\ & = & k^{\otimes^r}_{[\![\alpha]\!]}([\![\underline{\phi}']\!] \otimes^r [\![\psi']\!]) & \\ & = & k_{[\![\alpha]\!]}([\![\underline{\phi}']\!],[\![\psi']\!]) & \\ & = & [\![\underline{\phi}']\!] \otimes^r ([\![\alpha]\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\psi']\!]) & \\ & = & [\![\underline{\phi}']\!] \otimes^r [\![\alpha\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi']\!] & \text{by induction} \\ & = & \vee\ \{\, [\![\underline{\phi}']\!] \otimes^r [\![\xi]\!] \mid \xi \in \alpha\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi' \,\} & \text{since } i^{\otimes^r} \text{ is right-linear} \\ & = & \vee\ \{\, [\![[\underline{\phi}']\xi]\!] \mid \xi \in \alpha\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi' \,\} & \\ & = & \vee\ [\![S_2]\!]. & \end{array}$$

- Let k_x denote the function $\lambda(d,d') \in \mathbf{P} \times \mathbf{P}.x(m?)d\ {}_{\mathcal{A}}|_{\mathcal{B}}\ d'$. Then

$$\begin{array}{rcl} com^m_l([\![\alpha]\!],[\![\beta]\!]) & = & k^{\otimes^r}_{[\![\alpha]\!]}([\![\beta]\!](m!)) \\ & = & k^{\otimes^r}_{[\![\alpha]\!]}([\![\underline{\phi}']\!] \otimes^r [\![\psi]\!]) \\ & = & k_{[\![\alpha]\!]}([\![\underline{\phi}']\!],[\![\psi]\!]) \\ & = & ([\![\underline{\phi}]\!] \Rightarrow [\![\psi]\!])[\![\underline{\phi}']\!]\ {}_{\mathcal{A}}|_{\mathcal{B}}\ [\![\psi']\!] \end{array}$$

There are now two cases. If $[\![\underline{\phi}]\!] \leq [\![\underline{\phi'}]\!]$, i.e. $\mathcal{L} \vdash \underline{\phi'} \leq \underline{\phi}$, then this reduces to $[\![\psi]\!]\,_{\mathcal{A}}|_{\mathcal{B}}\,[\![\psi']\!]$ and by induction this coincides with $[\![S_3]\!]$. Moreover in this case, since $_{\mathcal{A}}|_{\mathcal{B}}$ is monotonic, $[\![S_3]\!] = [\![S_3]\!] \vee [\![S_4]\!]$. The second case is when $[\![\underline{\phi}]\!] \not\leq [\![\underline{\phi'}]\!]$ where a similar argument shows that the above expression reduces to $[\![S_4]\!]$.

□

This view of the the auxiliary functions in terms of formulae is also consistent with the realizability interpretation of processes given in section two. For convenience this is extended to sequences or sets of formulae by letting $p \models^{\mathcal{O}} S$ mean that $p \models^{\mathcal{O}} \phi$ for every formula $\phi \in S$.

Theorem 5.3 *For every G in Aux $P_i \models^{\mathcal{O}} \phi_i$ implies $G(\underline{P}) \models^{\mathcal{O}} G(\underline{\phi})$*

Proof: We use two easily established properties of the relation $\models^{\mathcal{O}}$, namely

1. $((X)T)P \models^{\mathcal{O}} \phi$ if and only if $T\{P/X\} \models^{\mathcal{O}} \phi$
2. If $P \stackrel{\tau}{\Longrightarrow} Q$ and $Q \models^{\mathcal{O}} \phi$ then $P \models^{\mathcal{O}} \phi$.

We consider briefly the proof of the theorem in the case when G is the parallel operator $_{\mathcal{A}}|_{\mathcal{B}}$. We must show that if $P_1 \models^{\mathcal{O}} \phi_1$ and $P_2 \models^{\mathcal{O}} \phi_2$ then $P_1\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \models^{\mathcal{O}} \eta$ for every $\eta \in \phi_1\,_{\mathcal{A}}|_{\mathcal{B}}\,\phi_2$. The proof proceeds by simultaneous induction on the structure of ϕ_1 and ϕ_2. As an example we consider the case when ϕ_1 is $\langle n?\rangle\underline{\phi} \rightarrow \psi$ and ϕ_2 is $\langle m!\rangle\underline{\phi'} \rightarrow \psi'$ where $n \in \mathcal{A}$, $m \in \mathcal{B}$ and $m = n$. Here there are five non-trivial possibilities for η.

- $\langle n?\rangle\underline{\phi} \rightarrow \xi$ where $\xi \in \psi\,_{\mathcal{A}}|_{\mathcal{B}}\,\beta$
  Since $P_1 \models^{\mathcal{O}} \phi_1$ it follows that $P_1 \stackrel{n?}{\Longrightarrow} (X)T$ for some $(X)T$ such that $(X)T \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$. By the operational semantics of $_{\mathcal{A}}|_{\mathcal{B}}$ it follows that $P_1\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \stackrel{n?}{\Longrightarrow} (X)(T\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2)$ and we show that $(X)(T\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2) \models^{\mathcal{O}} \underline{\phi} \rightarrow \xi$. Suppose that $Q \models^{\mathcal{O}} \underline{\phi}$; it is neccessary to show $((X)(T\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2))Q \models^{\mathcal{O}} \xi$, i.e. $T\{Q/X\}\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \models^{\mathcal{O}} \xi$. But since $(X)T \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$ and $Q \models^{\mathcal{O}} \underline{\phi}$ it follows that $T\{Q/X\} \models^{\mathcal{O}} \psi$. We may now use induction to conclude that $T\{Q/X\}\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \models^{\mathcal{O}} \xi$ since $\xi \in \psi\,_{\mathcal{A}}|_{\mathcal{B}}\,\beta$.

- $\langle n?\rangle\underline{\omega} \rightarrow \xi$ where $\xi \in \omega\,_{\mathcal{A}}|_{\mathcal{B}}\,\beta$
  Again we know that $P_1\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \stackrel{n?}{\Longrightarrow} (X)(T\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2)$ for some $(X)T$ such that $(X)T \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$. We must show $((X)(T\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2))Q \models^{\mathcal{O}} \xi$ for every Q since $Q \models^{\mathcal{O}} \omega$ for any Q. But $((X)T)Q \models^{\mathcal{O}} \omega$ and so by induction $((X)T)Q\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \models^{\mathcal{O}} \xi$. The result now follows by property 1 above.

- $\langle m!\rangle[\underline{\phi'}]\xi$ where $\xi \in \alpha\,_{\mathcal{A}}|_{\mathcal{B}}\,\psi'$
  Similar to the first case.

- $\eta \in \omega\,_{\mathcal{A}}|_{\mathcal{B}}\,\psi'$
  We know $P_1 \stackrel{n?}{\Longrightarrow} F$ such that $F \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$ and $P_2 \stackrel{n!}{\Longrightarrow} [Q_1]Q_2$ such that $Q_1 \models^{\mathcal{O}} \underline{\phi'}$ and $Q_2 \models^{\mathcal{O}} \psi'$. Since we are assuming $m = n$ $P_1\,_{\mathcal{A}}|_{\mathcal{B}}\,P_2 \stackrel{\tau}{\Longrightarrow} FQ_1\,_{\mathcal{A}}|_{\mathcal{B}}\,Q_2$ and so it is sufficient to prove that $FQ_1\,_{\mathcal{A}}|_{\mathcal{B}}\,Q_2 \models^{\mathcal{O}} \eta$. But since $FQ_1 \models^{\mathcal{O}} \omega$ this follows by induction.

- $\eta \in \psi\ {}_{\mathcal{A}}|_{\mathcal{B}}\ \psi'$ and $\mathcal{L} \vdash \underline{\phi'} \leq \underline{\phi}$
  As in the previous case we know $P_1\ {}_{\mathcal{A}}|_{\mathcal{B}}\ P_2 \stackrel{\tau}{\Longrightarrow} FQ_1\ {}_{\mathcal{A}}|_{\mathcal{B}}\ Q_2$ for some F, Q_1 and Q_2 such that $F \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$, $Q_1 \models^{\mathcal{O}} \underline{\phi'}$ and $Q_2 \models^{\mathcal{O}} \psi'$. We are assuming $\mathcal{L} \vdash \underline{\phi'} \leq \underline{\phi}$ and therefore, again by Proposition 4.6, $Q_1 \models^{\mathcal{O}} \underline{\phi}$ which implies in turn that $FQ_1 \models^{\mathcal{O}} \psi$. As in the previous case the result now follows by induction.

□

We end this section with a definability theorem: every prime, and therefore compact element, in $\mathbf{P}$ is definable by a term on our language. For each formula ϕ we construct a set of processes $P^{n,i}$, parameterised on pairs of distinct names n, i, and a set of closed abstraction terms $F^{n,i}$, parameterised in the same manner, such that if n, i does not appear in ϕ then

- $[\![P^{n,i}_{\phi}]\!] = [\![\phi]\!]$
- for all $d \in \mathbf{P}$, $[\![n!]\!] = [\![F^{n,i}_{\phi}]\!]d$ if and only if $[\![\phi]\!] \leq d$.

Moreover the abstractions $F^{n,i}$ have the form $(X)(T^{n,i}_{\phi}\ {}_{\{n\}}|_{\emptyset}\ X)$ for some process $T^{n,i}_{\phi}$ so that its application to a process is in fact the application of the test $T^{n,i}_{\phi}$. In order to define these terms we need some notation. For any pair of process terms T, U and name n let $T \rhd^n U$ denote the term $T\ {}_{\emptyset}|_{\mathcal{N}-\{n\}}\ n?(Y)U$ where U does not occur free in U. If $Y_1, \ldots, Y_k$ is a sequence of distinct variables let $con^{n,i}[Y_1, \ldots, Y_k]$ denote the term

$$Y_1[n \rightarrow i](\rhd^i Y_2[n \rightarrow i](\rhd^i \ldots (\rhd^i Y_k) \ldots))$$

where $[n \rightarrow i]$ is the renaming which sends n to i and is the identity elsewhere. Note that if $k = 1$ then $C[Y_1]$ is simply Y_1. We also use $T \backslash n$ to denote the term $NIL\ {}_{\emptyset}|_{\mathcal{N}-\{n\}}\ T$ and finally let W_n be the set consisting of two semantic elements, $\{\bot, [\![n!]\!]\}$. We leave the reader to check the following:

Lemma 5.4 *1. If $[\![P_i]\!] \in W_n$ then $[\![\ con^{n,i}[P_1, \ldots, P_k]]\!] \in W_n$*

2. If n does not occur in ϕ then $[\![\phi]\!] \backslash n = [\![\phi]\!]$. □

The definition of the required terms is by induction on the structure of formulae:

- ω
  $P^{n,i}_{\phi} = \bot$ and $F^{n,i}_{\phi} = (X)(n!\ {}_{\{n\}}|_{\emptyset}\ X)$
- $\langle m? \rangle \underline{\phi} \rightarrow \psi$
  $F^{n,i}_{\phi} = (X)(m![P^{n,i}_{\underline{\phi}}]T^{n,i}_{\psi}\ {}_{\{n\}}|_{\emptyset}\ X)$ and $P^{n,i}_{\phi} = m?(X)(F^{n,i}_{\underline{\phi}} X \rhd^n P^{n,i}_{\psi})$
- $\langle m! \rangle [\underline{\phi}] \psi$
  $F^{n,i}_{\phi} = (X)(m?(X)(F^{n,i}_{\underline{\phi}} X[n \rightarrow i] \rhd^i T^{n,i}_{\psi})\ {}_{\{n\}}|_{\emptyset}\ X)$ and $P^{n,i}_{\phi} = m![P^{n,i}_{\underline{\phi}}]P^{n,i}_{\psi}$.

where $P^{n,i}_{\underline{\phi}}$ denotes $P^{n,i}_{\phi_1} + \ldots + P^{n,i}_{\phi_k}$ and $F^{n,i}_{\underline{\phi}}$ the term $(X)\ con^{n,i}[F^{n,i}_{\phi_1} X, \ldots, F^{n,i}_{\phi_1} X]$.

Theorem 5.5 *(Definability) For any n, i not occurring in ϕ*

1. *for all $d \in \mathbf{P}$, $[\![n!]\!] = [\![F^{n,i}_{\phi}]\!]d$ if and only if $[\![\phi]\!] \leq d$*
2. $[\![P^{n,i}_{\phi}]\!] = [\![\phi]\!]$
3. *for all $d \in \mathbf{P}$, $[\![F^{n,i}_{\phi}]\!]d \in W_n$*
4. $[\![T^{n,i}_{\phi}]\!] \backslash i = T^{n,i}_{\phi}$.

Proof: By structural induction on formulae. As an example we consider the case when ϕ is the formula $\langle m! \rangle [\underline{\phi}]\psi$. There are four statements to prove.

1. Suppose $[\![n!]\!] = [\![F^{n,i}_{\phi}]\!]d$, i.e. $[\![n!]\!] = m_{in}([\![(X)F^{n,i}_{\underline{\phi}}X]\!][n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _{\{n\}}|_{\emptyset}\ d$. We must show that $m_{out}([\![\underline{\phi}]\!] \otimes^r [\![\psi]\!]) \leq d$. First notice that $d(m!) \neq \bot$ for otherwise $[\![F^{n,i}_{\phi}]\!]d$ would evaluate to $\bot$, contradicting the fact that $[\![n!]\!] \leq [\![F^{n,i}_{\phi}]\!]d$. Let $d(m!) = r_{\bot}$ where $r = \bigvee_J (p_j \otimes^r q_j)$ for some non-empty set J where each $p_j \otimes^r q_j$ is a prime. Then calculating $m_{in}([\![(X)F^{n,i}_{\underline{\phi}}X]\!][n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _{\{n\}}|_{\emptyset}\ d$ we obtain $(\lambda^{\otimes^r}(d_1, d_2) \in \mathbf{P} \times \mathbf{P}.([\![F^{n,i}_{\underline{\phi}}]\!]d_1[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!]\ _{\{n\}}|_{\emptyset}\ d_2))d(m!)$ which reduces to $\bigvee_J([\![F^{n,i}_{\underline{\phi}}]\!]p_j[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _n|\ q_j$ and since $[\![n!]\!]$ is prime it follows that $[\![n!]\!] \leq ([\![F^{n,i}_{\underline{\phi}}]\!]p_j[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _n|\ q_j$ for some j.

   We show that $[\![\underline{\phi}]\!] \leq [\![p_j]\!]$ and $[\![\psi]\!] \leq [\![q_j]\!]$ from which it immediately follows that $[\![\underline{\phi}]\!] \otimes^r [\![\psi]\!] \leq r$ and therefore that $[\![\langle m! \rangle [\underline{\phi}]\psi]\!] \leq d$.

   We must have that $[\![F^{n,i}_{\underline{\phi}}]\!]p_j[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!] \neq \bot$ which in turn dictates that $[\![F^{n,i}_{\underline{\phi}}]\!]p_j \neq \bot$. We know, by induction, that for each k $[\![F^{n,i}_{\phi_k}]\!]p_j \in W_n$. For no k can this evaluate to $\bot$ and therefore, since it evaluates to $[\![n!]\!]$, by induction we can conclude that $[\![\phi_k]\!] \leq [\![p_j]\!]$ for each k and so that $[\![\underline{\phi}]\!] \leq [\![p]\!]$.

   Notice that it follows from the previous Lemma that $[\![F^{n,i}_{\underline{\phi}}]\!]p_j$ is also in W_n and therefore it must evaluate to $[\![n!]\!]$. So we can calculate:

$$\begin{aligned} [\![F^{n,i}_{\underline{\phi}}]\!]p_j[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!] &= [\![n!]\!][n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!]) \\ &= [\![T^{n,i}_{\psi}]\!] \backslash i \\ &= [\![T^{n,i}_{\psi}]\!], \text{ by induction, part 4.} \end{aligned}$$

   This means that $[\![n!]\!] \leq ([\![T^{n,i}_{\psi}]\!])\ _{\{n\}}|_{\emptyset}\ q_j$ and by induction we therefore have that $[\![\psi]\!] \leq q_j$.

   Conversely suppose $m_{out}([\![\underline{\phi}]\!] \otimes^r [\![\psi]\!]) \leq d$, i.e. $[\![\underline{\phi}]\!] \otimes^r [\![\psi]\!] \leq r$ where $d(m!) = r_{\bot}$. We must show that $m_{out}([\![(X)F^{n,i}_{\underline{\phi}}X]\!] \rhd^n [\![T^{n,i}_{\psi}]\!])\ _{\{n\}}|_{\emptyset}\ d$ dominates $[\![n!]\!]$ for any n, i not occurring in ϕ. Again calculating $m_{out}([\![(X)F^{n,i}_{\underline{\phi}}X]\!][n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _n|\ d$ we obtain $\bigvee_J([\![F^{n,i}_{\underline{\phi}}]\!]p_j[n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _n|\ q_j$ where, as before, $r = \bigvee_J (p_j \otimes^r q_j)$. Since $[\![\underline{\phi}]\!] \otimes^r [\![\psi]\!]$ is prime there exists some j such that $[\![\underline{\phi}]\!] \leq p_j$ and $[\![\psi]\!] \leq q_j$. So $m_{out}([\![(X)F^{n,i}_{\underline{\phi}}X]\!][n \to i] \rhd^i [\![T^{n,i}_{\psi}]\!])\ _n|\ d$ dominates $([\![F^{n,i}_{\underline{\phi}}]\!]p_j \rhd^n [\![T^{n,i}_{\psi}]\!])\ _n|\ q_j$ for this

particular j. By induction we have that $[\![n!]\!] = [\![F^{n,i}_{\phi_l}]\!]p_j$ for each l and therefore, by calculation, $[\![n!]\!] = [\![F^{n,i}_{\underline{\phi}}]\!]p_j$. So

$$\begin{aligned}
[\![F_\phi]\!]d &\geq [\![i!]\!] \rhd^i [\![T^{n,i}_\psi]\!]\,_{\{n\}}|_\emptyset\, q_j \\
&= ([\![T^{n,i}_\psi]\!]\backslash i)\,_{\{n\}}|_\emptyset\, q_j \\
&= [\![T^{n,i}_\psi]\!]\,_{\{n\}}|_\emptyset\, q_j, \quad \text{by induction, part 4} \\
&= [\![n!]\!], \quad \text{by induction}
\end{aligned}$$

2. obvious by induction

3. By induction we know that $[\![F^{n,i}_{\phi_j}]\!]d \in W_n$ for each j and it follows by the previous Lemma that $[\![F^{n,i}_{\underline{\phi}}]\!]d \in W_n$. So $[\![F^{n,i}_{\phi}]\!]d$ has the form $g \rhd^i [\![T^{n,i}_\psi]\!]\,_{\{n\}}|_\emptyset\, d$ where $g \in W_i$. If g is $\perp$ then obviously this also reduces to $\perp$ which is in W_n. Otherwise g must be $[\![i!]\!]$ in which case it reduces to $([\![T^{n,i}_\psi]\!]\backslash i)\,_{\{n\}}|_\emptyset\, d$ which by induction, part 4, is the same as $[\![T^{n,i}_\psi]\!]\,_{\{n\}}|_\emptyset\, d$, i.e $[\![F^{n,i}_\psi]\!]d$. We can now apply induction to conclude that this is in W_n.

4.

$$\begin{aligned}
[\![T^{n,i}_\phi]\!]\backslash i &= (m_{in}\lambda d.[\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \rhd^i T^{n,i}_\psi)\backslash i \\
&= m_{in}\lambda d.([\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \rhd^i T^{n,i}_\psi)\backslash i, \\
&\qquad \text{by the definition of restriction.}
\end{aligned}$$

So it is sufficient to show that for any d

$$([\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \rhd^i T^{n,i}_\psi)\backslash i = [\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \rhd^i T^{n,i}_\psi. \qquad (*)$$

By induction and the previous Lemma we know that $[\![F^{n,i}_{\underline{\phi}}]\!]d \in W_n$ and therefore that $[\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \in W_i$. If this is in fact $\perp$ then obviously $(*)$ above is correct. Otherwise it must be $[\![i!]\!]$ in which case calculation gives

$$[\![F^{n,i}_{\underline{\phi}}]\!]d[n \to i] \rhd^i T^{n,i}_\psi = T^{n,i}_\psi$$

in which case $(*)$ follows by induction.

□

6 A Logical Characterisation of the Interpretation

In this section we show that the interpretation of the process language in the denotational model can also be captured in a logical form. We design a program logic whose judgements are of the form

$$\Gamma \vdash^a A : \phi$$

where $\phi \in \mathcal{L}^A$ for each $A = P, F, C$ and Γ is an assumption. A *assumption* is a finite map from $\mathcal{X}$ to non-empty finite subsets of $\mathcal{L}^P$. We will actually represent these finite subsets as vectors of the form $\underline{\phi}$. Let $\Gamma(X)$ denote ω whenever X is not in the domain of Γ and $\Gamma[X \mapsto \underline{\phi}]$ represent the assumption obtained from Γ in the obvious way – it coincides with Γ for every name except X which it maps to $\underline{\phi}$. We will also use $\Gamma \vdash^a A : \underline{\phi}$ as an abbreviation for the k judgements $\Gamma \vdash^a A : \phi_i$, $1 \leq i \leq k$ and by and large we omit the superscripts from consequence relations, rendering $\vdash^a$ as $\vdash$. The rules for deriving judgements are given in Figure 4.

The logic can be interpreted over the denotational model in the following manner. For $\sigma \in ENV$ let $\sigma \models \Gamma$ if $[\![\Gamma(X)]\!] \leq \sigma(X)$ for all X. Then we write $\Gamma \models^a A : \phi$ if $\sigma \models \Gamma$ implies $[\![\phi]\!] \leq [\![A]\!]\sigma$.

Proposition 6.1 *(Soundness)* $\Gamma \vdash^a A : \phi$ *implies* $\Gamma \models^a A : \phi$

Proof: We prove that the relation $\Gamma \models^a A : \phi$ satisfies the defining rules of $\Gamma \vdash^a A : \phi$. We give two cases.

1. The Rule FunR
   Suppose $\Gamma[X \mapsto \underline{\phi}] \models^p T : \psi$ and $\sigma \models \Gamma$. Then $\sigma[X \mapsto [\![\underline{\phi}]\!]] \models \Gamma[X \mapsto \underline{\phi}]$ and therefore $[\![\psi]\!] \leq [\![T]\!]\sigma[X \mapsto [\![\underline{\phi}]\!]]$. This means

   $$[\![\psi]\!] \leq (\lambda d \in \mathbf{P}.[\![T]\!]\sigma[X \mapsto d])[\![\underline{\phi}]\!],$$

   i.e.

   $$[\![\psi]\!] \leq ([\![(X)T]\!]\sigma)[\![\underline{\phi}]\!]$$

   i.e.

   $$([\![\underline{\phi}]\!] \Rightarrow [\![\psi]\!]) \leq [\![(X)T]\!]\sigma$$

2. The rule AuxR
   Suppose $\Gamma \vdash^p T_i : \phi_i$, $\sigma \models \Gamma$ and $\mathcal{L} \vdash G(\underline{\phi}) \leq \psi$. By induction we may assume that $[\![\phi_i]\!] \leq [\![T_i]\!]\sigma$. So

   $$\begin{aligned} [\![\psi]\!] &\leq [\![G(\underline{\phi})]\!] \quad \text{by part two of Theorem 4.4} \\ &= G([\![\phi_1]\!], \ldots, [\![\phi_k]\!]) \quad \text{by Theorem 5.2} \\ &\leq G([\![\underline{T}]\!]\sigma) \quad \text{since } G \text{ is monotonic} \\ &= [\![G(\underline{T})]\!]\sigma. \end{aligned}$$

□

This result also has a converse which is best expressed as a technical lemma. For any assumption Γ let σ_Γ be the environment defined by $\sigma_\Gamma(X) = [\![\Gamma(X)]\!]$.

Lemma 6.2 $[\![\phi]\!] \leq [\![A]\!]\sigma_\Gamma$ *implies* $\Gamma \vdash^a A : \phi$.

Proof: This time we use induction over A. We give three illustrative cases.

General :

$\mathcal{L}R$ $$\frac{\Gamma \vdash^a A : \underline{\phi}, \quad \mathcal{L} \vdash \underline{\phi} \leq \psi}{\Gamma \vdash^a A : \psi}$$

Processes :

NR $$\Gamma[X \mapsto \underline{\phi}] \vdash^p X : \phi_i$$

ωR $$\Gamma \vdash^p T : \omega$$

PreR $$\frac{\Gamma \vdash^f F : \phi}{\Gamma \vdash^p n?F : \langle n? \rangle \phi} \qquad \frac{\Gamma \vdash^c C : \phi}{\Gamma \vdash^p n!C : \langle n! \rangle \phi}$$

JoinR $$\frac{\Gamma \vdash^p T : \phi}{\Gamma \vdash^p T + U : \phi} \qquad \frac{\Gamma \vdash^p T : \phi}{\Gamma \vdash^p U + T : \phi}$$

ApR $$\frac{\Gamma \vdash^f F : \underline{\phi} \to \psi, \quad \Gamma \vdash^p T : \underline{\phi}}{\Gamma \vdash^p FT : \psi}$$

AuxR $$\frac{\Gamma \vdash^p T_i : \phi_i, \quad \mathcal{L} \vdash G(\underline{\phi}) \leq \psi}{\Gamma \vdash^p G(\underline{T}) : \psi}$$

Abstractions :

FunR $$\frac{\Gamma[X \mapsto \underline{\phi}] \vdash^p T : \psi}{\Gamma \vdash^f (X)T : \underline{\phi} \to \psi}$$

Concretions :

ConR $$\frac{\Gamma \vdash^p T : \underline{\phi}, \quad \Gamma \vdash^p U : \psi}{\Gamma \vdash^c [T]U : [\underline{\phi}]\psi}$$

Figure 4: The program logic

1. FT
   Suppose $[\![\psi]\!] \leq [\![FT]\!]\sigma_\Gamma = [\![F]\!]\sigma_\Gamma([\![T]\!]\sigma_\Gamma)$ where F has the form $(X)U$. Then

$$\begin{aligned} [\![\psi]\!] &\leq (\lambda d \in \mathbf{P}.[\![U]\!](\sigma_\Gamma[X \mapsto d]))([\![T]\!]\sigma_\Gamma) \\ &= [\![U]\!]\sigma_\Gamma[X \mapsto [\![T]\!]\sigma_\Gamma] \\ &= \bigvee \{\, [\![U]\!]\sigma_\Gamma[X \mapsto c] \mid c \leq [\![T]\!]\sigma_\Gamma \,\} \end{aligned}$$

Since $[\![\psi]\!]$ is compact there exists some $c \leq [\![T]\!]\sigma_\Gamma$ such that $[\![\psi]\!] \leq [\![U]\!]\sigma_\Gamma[X \mapsto c]$. Moreover there is a finite set $\underline{\phi}$ such that $c = [\![\underline{\phi}]\!]$ in which case $\sigma_\Gamma[X \mapsto c] = \sigma_{\Gamma[X \mapsto \underline{\phi}]}$. So $[\![\psi]\!] \leq [\![U]\!]\sigma_{\Gamma[X \mapsto \underline{\phi}]}$ and by induction $\Gamma[X \mapsto \underline{\phi}] \vdash U : \psi$. Applying the rule FunR we obtain $\Gamma \vdash (X)U : (\underline{\phi} \to \psi)$. Also $[\![\phi_i]\!] \leq [\![T]\!]\sigma_\Gamma$ for each ϕ_i and so by induction $\Gamma \vdash T : \phi_i$. An instance of the application rule now gives $\Gamma \vdash FT : \psi$.

2. $G(\underline{T})$
Suppose $[\![\psi]\!] \leq [\![G(\underline{T})]\!]\sigma_\Gamma = G([\![\underline{T}]\!]\sigma_\Gamma)$. Since $[\![\psi]\!]$ is prime and G is multilinear there exists a vector of primes $\underline{p}$ such that $p_i \leq [\![T_i]\!]\sigma_\Gamma$ and $[\![\psi]\!] \leq G(\underline{p})$. Let p_i be denoted by ϕ_i. Then $[\![\psi]\!] \leq G([\![\phi_1]\!], \ldots, [\![\phi_k]\!]) = [\![G(\underline{\phi})]\!]$. By the completeness of $\mathcal{L}$ we have $\mathcal{L} \vdash G(\underline{\phi}) \leq \psi$ and since $[\![\phi_i]\!] \leq [\![T_i]\!]\sigma_\Gamma$ we may apply induction to obtain $\Gamma \vdash T_i : \phi_i$. We now have the required premises of the rule AuxR which when applied gives $\Gamma \vdash G(\underline{T}) : \psi$.

3. $(X)T$
Suppose $[\![\underline{\phi} \to \psi]\!] \leq [\![(X)T]\!]\sigma_\Gamma$, i.e. $[\![\underline{\phi}]\!] \Rightarrow [\![\psi]\!] \leq \lambda d \in \mathbf{P}.[\![T]\!]\sigma_\Gamma[X \mapsto d]$. This means

$$\begin{aligned} [\![\psi]\!] &\leq [\![T]\!]\sigma_\Gamma[X \mapsto [\![\underline{\phi}]\!]] \\ &= [\![T]\!]\sigma_{\Gamma[X \mapsto \underline{\phi}]} \end{aligned}$$

We may now apply induction to obtain $\Gamma[X \mapsto \underline{\phi}] \vdash T : \psi$ and an application of the rule FunR gives $\Gamma \vdash (X)T : \underline{\phi} \to \psi$.

□

As an immediate corollary we have

Theorem 6.3 *(Completeness of Program Logic)* $\Gamma \models^a A : \phi$ *implies* $\Gamma \vdash^a A : \phi$

Proof: Follows because $\sigma_\Gamma \models \Gamma$ □

Thus this program logic captures the denotational semantics precisely. For let

$$\mathcal{L}_\Gamma(A) = \{\, \phi \mid \Gamma \vdash A : \phi \,\}.$$

Then

Theorem 6.4 $[\![A]\!] \leq [\![A']\!]$ *if and only if* $\mathcal{L}_\Gamma(A) \subseteq \mathcal{L}_\Gamma(A')$

Proof: Suppose $[\![A]\!] \leq [\![A']\!]$ and $\Gamma \vdash A : \phi$. Then by soundness, since $\sigma_\Gamma \models \Gamma$, it follows that $[\![\phi]\!] \leq [\![A]\!]\sigma_\Gamma$. So $[\![\phi]\!] \leq [\![A']\!]\sigma_\Gamma$ and by completeness $\Gamma \vdash A' : \phi$.
The converse is equally straightforward. □

We can also interpret the program logic using the realizability or observational interpretation of section two. Soundness is straightforward but the more difficult completeness will be given in the next section.

If ρ is a closed substitution we write $\rho \models^{\mathcal{O}} \Gamma$ if for every $X \in \mathcal{X}$ $\rho(X) \models^{\mathcal{O}} \Gamma(X)$ and $\Gamma \models^{\mathcal{O}} A : \phi$ if $\rho \models^{\mathcal{O}} \Gamma$ implies $A \models \phi$. Note that $\Gamma \models^{\mathcal{O}} A : \phi$ refers to the operational behaviour of A while $\Gamma \models A : \phi$ refers to its denotational interpretation.

Proposition 6.5 *$\Gamma \vdash A : \phi$ implies $\Gamma \models^{\mathcal{O}} A : \phi$.*

Proof: Again it is sufficient to prove that the relation $\Gamma \models^{\mathcal{O}} A : \phi$ satisfies all the defining rules of $\Gamma \vdash A : \phi$. This is relatively straightforward; the rule $\mathcal{L}R$ follows from Proposition 4.6 while rule AuxR also requires Proposition 5.3. □

7 Full Abstraction

In this section we prove the main results of the paper, connecting the denotational interpretation with the various behavioural preorders.

First a lemma which shows that the denotational interpretation is consistent with the operational semantics.

Lemma 7.1 *1. For c of the form $m?$ or $m!$, $P \xrightarrow{c} A$ implies $[\![A]\!]_\perp \leq [\![P]\!](c)$*

2. $P \xrightarrow{\tau} Q$ implies $[\![Q]\!] \leq [\![P]\!]$.

Proof: Both statements are proved simultaneously by induction on the length of the derivation of the operational judgements. □

As an immediate corollary we have the following *adequacy* result:

Theorem 7.2 *(Adequacy) For c of the form $m?$ or $m!$ $P \stackrel{c}{\Longrightarrow}$ if and only if $[\![P]\!](c) \neq \perp$.*

Proof: First suppose that $P \stackrel{c}{\Longrightarrow}$, i.e. $P \stackrel{\varepsilon}{\Longrightarrow} P' \xrightarrow{c} A$. The first part of the previous lemma implies $[\![P']\!](c) \neq \perp$ while the second part implies $[\![P']\!] \leq [\![P]\!]$, i.e. $[\![P]\!](c) \neq \perp$.

Conversely suppose $[\![P]\!](c) \neq \perp$. Then by the completeness theorem for the program logic, Theorem 6.3, $\vdash P : \langle c \rangle \omega$. By the soundness of the program logic with respect to the realizability interpretation, Proposition 6.5, $\models^{\mathcal{O}} P : \langle c \rangle \omega$, i.e. $P \stackrel{c}{\Longrightarrow}$. □

Note the crucial role played by the completeness of the program logic in establishing this result. Without such a characterisation of the denotational interpretation of the language this adequacy result would be very difficult to establish. Note also that, using the definition of $\Longrightarrow$ from section two, this result could also be stated as

$$P \Longrightarrow \quad \text{if and only if} \quad [\![P]\!] \neq \perp.$$

As an immediate corollary we have one direction of the full abstraction result:

Corollary 7.3 *For closed terms $[\![P]\!] \leq [\![Q]\!]$ implies $P \sqsubseteq_{\mathcal{O}} Q$.*

Proof: Let $C[\ ]$ be any closed context such that $C[P] \stackrel{c}{\Longrightarrow}$. We must show that $C[Q] \stackrel{c}{\Longrightarrow}$ which, by the adequacy result is equivalent to showing $[\![C[Q]]\!](c) \neq \bot$. But $[\![P]\!] \leq [\![Q]\!]$ implies $[\![C[P]]\!] \leq [\![C[Q]]\!]$ and another application of the adequacy result gives $[\![C[P]]\!](c) \neq \bot$. It therefore follows that $[\![C[Q]]\!](c) \neq \bot$. □

The definability result, Theorem 5.5, together with adequacy gives a converse

Proposition 7.4 *For closed terms* $P \ \underset{\sim}{\sqsubset}_{\mathcal{I}} \ Q$ *implies* $[\![P]\!] \leq [\![Q]\!]$.

Proof: To prove $[\![P]\!] \leq [\![Q]\!]$ it is sufficient to prove $[\![\phi]\!] \leq [\![Q]\!]$ for any ϕ such that $[\![\phi]\!] \leq [\![P]\!]$. For such a ϕ the definability theorem implies that $[\![n!]\!] \leq [\![F^{n,i}_{\phi}]\!][\![P]\!]$, for all n, i not appearing in ϕ, which can be rewritten as $[\![F^{n,i}_{\phi}]\!][\![P]\!](n!) \neq \bot$. By the adequacy theorem this implies $F^{n,i}_{\phi} P \stackrel{n!}{\Longrightarrow}$. In general $F^{n,i}_{\phi} R \stackrel{n!}{\Longrightarrow}$ if and only if $T^{n}_{\phi\ \{n\}|\emptyset}\ R \stackrel{n!}{\Longrightarrow}$ and since $P \ \underset{\sim}{\sqsubset}_{\mathcal{I}} \ Q$ it follows that $F^{n,i}_{\phi} Q \stackrel{n!}{\Longrightarrow}$ which, again by the adequacy theorem, implies $[\![n!]\!] \leq [\![F^{n,i}_{\phi}]\!][\![Q]\!]$. Employing the definability result once more we obtain $[\![\phi]\!] \leq [\![Q]\!]$. □

As an immediate corollary we have

Corollary 7.5 *(Full Abstraction) For closed terms* $P \ \underset{\sim}{\sqsubset}_{\mathcal{O}} \ Q$ *if and only if* $P \ \underset{\sim}{\sqsubset}_{\mathcal{I}} \ Q$ *if and only if* $P \ \underset{\sim}{\sqsubset}_{\mathcal{L}} \ Q$ *if and only if* $[\![P]\!] \leq [\![Q]\!]$.

Proof: One can easily establish that $\underset{\sim}{\sqsubset}_{\mathcal{O}}$ is contained in $\underset{\sim}{\sqsubset}_{\mathcal{I}}$ and therefore Corollary 7.3 and Proposition 7.4 establish that the both of these coincide with the preorder generated by the model. Because of Proposition 6.5 it is therefore sufficient to show $P \models^{\mathcal{O}} \phi$ implies $\vdash P : \phi$. By the completeness result Theorem 6.3 this is equivalent to showing $P \models^{\mathcal{O}} \phi$ implies $[\![\phi]\!] \leq [\![P]\!]$. The more general result $A \models^{\mathcal{O}} \phi$ implies $[\![\phi]\!] \leq [\![A]\!]$ for A any closed process, abstraction or concretion term, can be established by induction on ϕ. We give two cases:

- ϕ has the form $\langle c \rangle \psi$.
  In this case there exists some A such that $P \stackrel{c}{\Longrightarrow} A$ and $A \models^{\mathcal{O}} \psi$. By induction $[\![\psi]\!] \leq [\![A]\!]$ and by Lemma 7.1 $[\![A]\!]_{\bot} \leq [\![P]\!](c)$. It follows that $[\![\langle c \rangle \psi]\!] \leq [\![P]\!]$.

- ϕ has the form $\underline{\phi} \rightarrow \psi$.
  Here we use the fact that $[\![P^{n,i}_{\underline{\phi}}]\!] = [\![\underline{\phi}]\!]$ for any n, i not occurring in $\underline{\phi}$ and therefore, by Proposition 6.5, that $P^{n,i}_{\underline{\phi}} \models^{\mathcal{O}} \underline{\phi}$.

  Now suppose $F \models^{\mathcal{O}} \underline{\phi} \rightarrow \psi$. It follows by the definition of $\models^{\mathcal{O}}$ that $F P^{n,i}_{\underline{\phi}} \models^{\mathcal{O}} \underline{\psi}$ and therefore by induction that $[\![\psi]\!] \leq [\![F P^{n,i}_{\underline{\phi}}]\!]$. This means $[\![\psi]\!] \leq [\![F]\!]([\![\underline{\phi}]\!])$ and therefore that $[\![\underline{\phi} \rightarrow \psi]\!] \leq [\![F]\!]$.

□

8 Conclusions

We have presented a semantic model of higher–order processes and shown it to be fully abstract with respect to a number of observational preorders. But these results raise many questions, some quite specific about our technical development and others more general.

It has been shown in [San92] that higher–order process languages can be simulated in the π-calculus but this is not to say that such languages are superfluous. They may provide convenient specification formalisms at an appropriate level of abstraction for describing the behaviour of sophisticated systems such as distributed operating or control systems, [LB92]. If this is the case then what kind of combinators should such a language have and can we model them using this semantic domain? Another question concerns the channel scoping mechanism used in the language. As we have seen $\mathbf{P}$ is adequate for modelling dynamic scoping but it has been argued in [Tho90] that static scoping of channels leads to a more natural language. At the moment it is not clear how to amend the definition of $\mathbf{P}$ so as to correctly model static scoping.

The program logic presented in section 6 provides the theoretical basis for a proof system for deriving properties of higher–order processes. As already pointed out this logic is not very realistic as to prove a parallel process $p \ {}_{\mathcal{A}}|_{\mathcal{B}} \ q$ has a property ψ it is necessary to find two formulae ϕ_1, ϕ_2 such that we can prove that p has the property ϕ_1 and q ϕ_2 and then prove that the characteristic formula $\phi_1 \ {}_{\mathcal{A}}|_{\mathcal{B}} \ \phi_2$ logically implies ψ. However this rule $AuxR$ could, at least in the case of the parallel operator, be replaced by more useful or easily applicable rules. The exact form these replacement rules should take remains to be seen.

Another line of possible future research concerns the behavioural preorders being modelled. That studied in the present paper is very weak, at least compared to many preorders defined for first–order process calculi. For such calculi it essentially corresponds to trace inclusion: $P \leq Q$ if every sequence of actions which P can perform can also be performed by Q. This is very weak as it does not take into consideration the possible deadlocks or divergences of processes. For example it does not distinguish between the process $a.P$ and $a.P + a.NIL$ or $a.P + a.\Omega$ where Ω represents some process which can only perform an infinite internal computation. There are a large number of more discriminating behavioural preorders and equivalences in the extensive literature on process algebras, for example bisimulation equivalence [Mil89], failures equivalence [Hoa85] and the testing preorders of [Hen88]. These may easily be extended to higher–order processes and the approach to behavioural preorders in the definition of $\sqsubseteq_{\mathcal{O}}$ can also be strengthened so as to include information on deadlock. The simplest modification is to base the basic comparison $\prec$ between processes not on their ability to perform actions but on their ability to converge: $P \Downarrow$ if there is no infinite internal computation from P, $P \xrightarrow{\tau} P_i \xrightarrow{\tau} \ldots \xrightarrow{\tau}$ and $P \ may \Downarrow$ if there exists some Q such that $P \stackrel{\varepsilon}{\Longrightarrow} Q$ and $Q \Downarrow$. The basic comparison could now be defined by

$P \prec Q$ if $P \ may \Downarrow$ implies $P \ may \Downarrow$.

The resulting behavioural preorder is different than that which we have studied as it differentiates NIL from Ω whereas they are identified in our theory. An even stronger comparison could be defined by

$P \prec Q$ if $P \Downarrow$ implies $Q \Downarrow$.

This leads to a behavioural theory which in general differentiates between processes of the form $a.P$, $a.P + a.NIL$ and $a.P + a.\Omega$. It remains to be seen if fully abstract denotational models can be constructed for these theories.

Higher–order process calculi have been studied in a number of papers. In [Tho89, Tho90] the language CHOCS, on which our language is based, and a statically scoped version called Plain CHOCS are studied in detail. The theory of strong bisimulation equivalence is developed for these languages along the lines outlined in [AAR88] and a denotational model for CHOCS is presented which is fully abstract with respect to a modified version of strong bisimulation equivalence. Higher–order process calculi are also studied in [San92] where the main concern is their relationship with the π-calculus.

In [Bou89] a generalisation of the λ-calculus, called the γ-calculus, is defined in which a form of parallelism is allowed. A very restricted subset of this language is modelled in [JP90] using a new form of powerdomain construction. This model is shown to be adequate with respect to an operational semantics but it is not known if it is fully abstract. The addition of parallelism to the λ-calculus is studied extensively in [Bou90b, Bou91]; in particular fully-abstract models, filter models of logics, are constructed for the observational preorder over parallel-λ-terms. More recently Boudol has developed a language of communicating objects, [Bou92], for which he has obtained similar results. This language bears some similarity with our higher–order process language and the exact relationship warrants further investigation.

Other approaches to higher–order processes may be found in [AR87, GMP90, Nie89]. The overall aim of this work is the development of more realistic higher–order programming languages which contains among other things a sophisticated type structures for the values transmitted between processes.

Acknowledgements: Thanks to Gerard Boudol for his detailed comments on a first draft of this paper.

References

[AAR88] E. Astesiano, A.Giovini, and G. Reggio. Generalised bisimulation in relational specifications. In *Proceedings of STACS 88*, volume 294 of *Lecture Notes in Computer Science*, pages 207–226, 1988.

[Abr90] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison-Wesley, 1990.

[Abr91] Samson Abramsky. Domain theory in logical form. *Ann. Pure Appl. Logic*, 51:1–77, 1991.

[AO89] S. Abramsky and C. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 1989. to appear.

[AR87] E. Astesiano and G. Reggio. SMoLS-driven concurrent calculi. In *TAPSOFT 1987,* Lecture Notes in Computer Science 351, Lecture Notes in Computer Science, pages 169–201, 1987.

[Bar84] Henk Barendregt. *The Lambda Calculus.* North-Holland, 1984. Studies in logic 103.

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter model and the completeness of type assignment. *J. of Symbolic Logic*, 48:931–940, 1983.

[Bou89] G. Boudol. Towards a lambda–calculus for concurrent and communicating systems. In J. Diaz, editor, *Proc. TAPSOFT 89*, pages 149–161. Springer-Verlag, 1989. LNCS 351.

[Bou90a] G. Boudol. Flow event structures and flow nets. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, volume 469 of *Lecture Notes in Computer Science*, pages 62–95, 1990.

[Bou90b] G. Boudol. A lambda-calculus for parallel functions. Technical Report 1231, INRIA-Sophia Antipolis, 1990.

[Bou91] G. Boudol. Lambda-calculi for (strict) parallel functions. Technical Report 1387, INRIA-Sophia Antipolis, 1991. To appear in *Information and Computation.*

[Bou92] G. Boudol. A calculus of communicating objects, 1992. To appear as INRIA Research Report.

[BW90] J. Baeton and W. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Computer Science.* Cambridge University Press, 1990.

[CC90] F. Cardone and M. Coppo. Two extensions of curry's type inference system. In P. Odifreddi, editor, *Logic in Computer Science*, pages 19–25. Academic Press, 1990.

[DH84] R. DeNicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 24:83–113, 1984.

[GMP90] A. Giacalone, P. Mistra, and S. Prasad. Operational and algebraic semantics for facile: A symmetric integration of concurrent and functional programming. In *Proceedings of ICALP 90*, volume 443 of *Lecture Notes in Computer Science*, pages 765–780, 1990.

[Gue81] I. Guessarian. *Algebraic Semantics.* Lecture Notes in Computer Science vol 99, 1981.

[Hen88] M. Hennessy. *An Algebraic Theory of Processes.* MIT Press, 1988.

[HI91] M. Hennessy and A. Ingolfsdottir. A theory of communicating processes with value-passing. *Information and Computation*, to appear, 1991.

[Hin83] R. Hindley. The completeness theorem for typing λ-terms. *Theoretical Computer Science*, 22:1–7 and 127–133, 1983.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, 1985.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[JP90] R. Jagadeesan and P. Panangaden. A domain–theoretic model for a higher–order process calculus. In M.S.Paterson, editor, *Proc. ICALP 90*, pages 181–194. Springer-Verlag, 1990. LNCS 443.

[LB92] L. Leth and B.Thomsen. Some facile chemistry. Technical Report ERCC-92-14, ERCC, 1992.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[Mil91] Robin Milner. The polyadic π-calculus: a tutorial. In *Proc. International Summer School on Logic and Algebra of Specification*, Marktoberdorf, 1991.

[MPW92a] R. Milner, J. Parrow, and D. Walker. A calculus of mobile proceses, part i. *Information and Computation*, 100(1):1–40, 1992.

[MPW92b] R. Milner, J. Parrow, and D. Walker. A calculus of mobile proceses, part ii. *Information and Computation*, 100(1):41–77, 1992.

[Nie89] F. Nielson. The typed λ-calculus with first–class processes. In *Proceedings of Parle 89*, volume 366 of *Lecture Notes in Computer Science*, 1989.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[San92] D. Sangiorgo. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* Phd thesis, Edinburgh University, Scotland, 1992.

[Sco82] D. S. Scott. Domains for denotational semantics. In M. Neilsen and E. M. Schmidt, editors, *Proc. ICALP 82*, pages 577–613. Springer-Verlag, 1982. LNCS 140.

[Tho89] B. Thomsen. A calculus of higher order communicating systems. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages*, pages 143–154, 1989.

[Tho90] B. Thomsen. *Calculi for Higher-Order Communicating Systems.* Phd thesis, Imperial College, 1990.