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, $\mid$ 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 $c!Q.P \quad \mid \quad c?X.(X \mid R)$ 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 $\lambda x dsim Q stract$

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

Each λ-term is interpreted either as $\perp$, 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 $\mathbf{F}_\perp$; with respect to each channel the process may offer no behaviour, modelled by $\perp$, 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 $\mathbf{C}_\perp$, 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 $\mathbf{C} \quad {}^{r}\, \mathbf{C}$.

The analogy

In [Abr90] it is shown that, subject to certain expressivity requirements, the domain $\mathbf{D}$ is fully abstract with respect to the observational preorder $\underset{\sim}{\sqsubset}_{\mathcal{O}}$. That is, $p \underset{\sim}{\sqsubset}_{\mathcal{O}} q$ 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 $\underset{\sim}{\sqsubset}_{\mathcal{O}}$, 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:

> $p \underset{\sim}{\sqsubset}_{\mathcal{T}} q$ 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. $\underset{\sim}{\sqsubset}_{\mathcal{O}}$ and $\underset{\sim}{\sqsubset}_{\mathcal{T}}$ 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 $\underset{\sim}{\sqsubset}_{\mathcal{T}}$ to higher-order processes by saying p *may satisfy* the test e if there is a successful application of e to p and

> $p \underset{\sim}{\sqsubset}_{\mathcal{T}} q$ 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 $\underset{\sim}{\sqsubset}_{\mathcal{T}}$.

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 **D** is isomorphic to the filters generated by the logic. Further the interpretation of the λ-calculus in **D** can be completely captured by a program logic whose judgements are of the form $\vdash p$

- $P \xrightarrow{\tau} Q$: This means

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 $\perp = \bigvee \emptyset$ 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 $f\colon D \longmapsto E$ between two domains is *strict* if $f(\perp) = \perp$, *monotonic* if $d \leq d'$

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 $in_\perp \colon D \longmapsto D_\perp$ 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 $f \colon D_1 \times D_2 \longmapsto E$ 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 $D_1 \quad {}^r D_2$ be the domain characterised by the requirements

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

$$\begin{tikzcd}
D_1 \times D_2 \arrow[d, "i^{\otimes^r}"] \arrow[dr, "f"] & \\
D_1 \quad {}^r D_2 \arrow[r, "f^{\otimes^r}"] & E
\end{tikzcd}$$

Of course we have to show that such a $D_1 \quad {}^r D_2$ 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 $D_1 \quad {}^r D_2$ 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 $D_1 \quad {}^r\, D_2$ be $\mathcal{P}_l(P)$. Let $i\colon \mathcal{K}(D_1 \times D_2) \longmapsto D_1 \quad {}^r\, D_2$ 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 $D_1 \quad {}^r\, D_2$. Note that this is well defined for if $p_i,\ qD$

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

$$\begin{aligned} P &\xrightarrow{n?} F \\ P &\xrightarrow{n!} C \\ P &\xrightarrow{\tau} Q, \end{aligned}$$

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

τ

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 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 different areas of research. On the one

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 $m![\ n!\ ]NIL + m![\ \overline{k!}\ ]NIL$ 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 $P_2 \not\lesssim_{\mathcal{L}} Q_2$.

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

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

Intuitively this models a process using two

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

General :

Refl $$\phi \leq \phi$$

Weak $$\frac{\underline{\phi} \leq \psi}{\underline{\phi},\ \phi' \leq \psi}$$

Trans $$\frac{\underline{\phi} \leq \underline{\psi},\quad \underline{\psi} \leq \xi}{\underline{\phi} \leq \xi}$$

Processes :

$\mathcal{L}P_1$ $$\phi \leq \omega$$

$\mathcal{L}P_2$ $$\frac{\phi \leq \psi}{\langle c \rangle \phi \ \leq \ \langle c \rangle \psi}$$

Abstractions :

$\mathcal{L}F_1$ $$\phi \leq (\omega \rightarrow \omega)$$

$\mathcal{L}F_2$ $$\frac{\underline{\phi} \leq \underline{\phi}',\quad \psi \leq \psi'}{\underline{\phi}}$$

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

Definition 4.3

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

□

Using this interpretation

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

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

2. $\mathcal{L} \vdash \underline{\phi} \leq \underline{\psi}$ *if and only if* $\llbracket \underline{\psi} \rrbracket \leq \llbracket \underline{\phi} \rrbracket$.

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 $\llbracket \phi \rrbracket = p$. 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 $\llbracket \psi \rrbracket = f$. It follows that $\llbracket \langle n? \rangle \psi \rrbracket = p$.

The proof that $\mathcal{L} \vdash \underline{\phi} \leq \psi$ implies $\llbracket \psi \rrbracket \leq \llbracket \underline{\phi} \rrbracket$ 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 $\llbracket \underline{\psi} \rrbracket \leq \llbracket \underline{\phi} \rrbracket$. We prove the converse and it is sufficient to prove $\llbracket \psi \rrbracket \leq \llbracket \phi \rrbracket$ implies $\mathcal{L} \vdash \phi \leq \psi$. For suppose we have established this and that $\llbracket \underline{\psi} \rrbracket \leq \llbracket \underline{\phi} \rrbracket$. This means that $\llbracket \psi_i \rrbracket \leq \llbracket \underline{\phi} \rrbracket$ for each i and since $\llbracket \psi_i \rrbracket$ is prime this implies $\llbracket \psi_i \rrbracket \leq \llbracket \phi_j \rrbracket$ 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 $\llbracket \psi \rrbracket \leq \llbracket \phi \rrbracket$ 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 $\llbracket \psi \rrbracket \leq \llbracket \phi \rrbracket$ it follows that ϕ must be of the form $\langle n? \rangle \xi$; otherwise $\llbracket \phi \rrbracket (n?) = \bot$ and so $\llbracket \phi \rrbracket$ would not dominate $\llbracket \psi \rrbracket$. Moreover it is easy to check that $\llbracket \eta \rrbracket \leq \llbracket \xi \rrbracket$ 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.

. $\psi = [\underline{\psi}^1] \psi^2$
   Let ϕ have the form $[\underline{\phi}^1] \phi^2$. So $\llbracket \underline{\psi}^1 \rrbracket \quad {}^{r} \llbracket \psi^2 \rrbracket \leq \llbracket \underline{\phi}^1 \rrbracket \quad {}^{r} \llbracket \phi^2 \rrbracket$ 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) $[\![\psi^2]\!] = \perp$.

   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 $\mathcal{L}F_2$, $\mathcal{L}$

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:

- $[\![NIL]\!]_P\sigma = \bot$
- $[\![n?F]\!]_P\sigma = n_{in}([\![F]\!]_F\sigma)$
- $[\![n!C]\!]_P\sigma = n_{out}([\![C]\!]_C\sigma)$
- $[\![X]\!]_P\sigma = \sigma(X)$
- $[\![T+U]\!]_P\sigma = [\![T]\!]_P\sigma \vee [\![U]\!]_P\sigma$
- $[\![FT]\!]_P\sigma = [\![F]\!]_F\sigma([\![T]\!]_P\sigma)$
- $[\![(X)T]\!]_F\sigma = \lambda d \in \mathbf{P}.[\![T]\!]_P\sigma[X \mapsto d]$
- $[\![[T]U]\!]_C\sigma = [\![T]\!]_P\sigma \quad {}^{r}\, [\![U]\!]_P\sigma$
- $[\![G(\underline{T})]\!]_P\sigma = g([\![\underline{T}]\!]_P)$
  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 $[D_1 \quad {}^{r}\, D_2 \longrightarrow E]$.

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

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

$k_{y\vee y'}^{\otimes^r} = k_y^{\otimes^r} \vee k_{y'}^{\otimes^r}$. Therefore

$$\begin{array}{rcl} f_{out}^m(x, y \vee y') & = & k_{y\vee y'}^{\otimes^r} x(m!) \\ & = & k_y^{\otimes^r} x(m!) \quad \vee \quad k_{y'}^{\otimes^r} x(m!) \\ & = & f_{out}^m(x,y) \vee f_{out}^m(x,y'). \end{array}$$

- com_l^m is multilinear.
  Here let k_z denote the function $\lambda(d, d') \in \mathbf{P} \times \mathbf{P}.F(z(m$

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

Figure : 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 α, β 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

$$
\begin{array}{rcl}
S_{11} & = & \{\, \underline{\phi} \rightarrow \xi \mid \xi \in \psi \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \beta \,\} \\
S_{12} & = & \{\, \omega \rightarrow \xi \mid \xi \in \omega \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \beta \,\} \\
S_{2} & = & \{\, [\underline{\phi}]\xi \mid \xi \in \alpha \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \psi' \,\} \\
S_{3} & = & \{\, \xi \mid \xi \in \psi \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \psi',\ \mathcal{L} \models \underline{\phi}' \leq \underline{\phi} \} \\
S_{4} & = & \{\, \xi \mid \xi \in \omega \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \psi' \,\}
\end{array}
$$

To prove the result it is therefore sufficient to establish

$$f$$

There are

- $\eta \in \psi \,{}_{\mathcal{A}}|_{\mathcal{B}}\, \psi'$ and $\mathcal{L} \vdash \underline{\phi'} \leq \underline{\phi}$
  As in the previous case we know $P_1 \,{}_{\mathcal{A}}|_{\mathcal{B}}\, P_2 \overset{\tau}{\Longrightarrow} FQ_1 \,{}_{\mathcal{A}}|_{\mathcal{B}}\, Q_2$ 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 $F^{n,i}_{\phi} = (X)(n! \,{}_{\{n\}}|_{\emptyset}\, X)$
- $\langle m? \rangle \phi$

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

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

Proof: By structural induction on formulae. As an example we consider the case when ϕ is the formula $\langle m! \rangle [\phi$

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

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

2. obvious by induction

. By induction we know that $\llbracket F^{n,i}_{\phi_j} \rrbracket d \in W_n$ for each j and it follows by the previous Lemma that $\llbracket F^{n,i}_{\underline{\phi}} \rrbracket d \in W_n$. So $\llbracket F^{n,i}_\phi \rrbracket d$ has the form $g \rhd^i \llbracket T^{n,i}_\psi \rrbracket \,_{\{n\}}|_\emptyset \, d$ 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 $\llbracket i! \rrbracket$ in which case it reduces to $(\llbracket T^{n,i}_\psi \rrbracket \backslash i)$

$\emptyset$

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 $\llbracket \psi \rrbracket \leq \llbracket FT \rrbracket \sigma_\Gamma = \llbracket F \rrbracket \sigma_\Gamma(\llbracket T \rrbracket \sigma_\Gamma)$ where F has the form $(X)U$. Then

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

Since $\llbracket \psi \rrbracket$ is compact there exists some $c \leq \llbracket T \rrbracket \sigma_\Gamma$ such that $\llbracket \psi \rrbracket \leq \llbracket U \rrbracket \sigma_\Gamma[X \mapsto c]$. Moreover there is a finite set $\underline{\phi}$ such that $c = \llbracket \underline{\phi} \rrbracket$ in which case $\sigma_\Gamma[X \mapsto c] = \sigma_{\Gamma[X \mapsto \underline{\phi}]}$. So $\llbracket \psi \rrbracket \leq \llbracket U \rrbracket \sigma_{\Gamma[X \mapsto \underline{\phi}]}$ and by induction $\Gamma[X \mapsto \underline{\phi}] \vdash U : \psi$. Applying the rule FunR we obtain $\Gamma \vdash (X)U : (\underline{\phi} \rightarrow \psi)$. Also $\llbracket \phi_i \rrbracket \leq \llbracket T \rrbracket \sigma_\Gamma$ 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 $\llbracket \psi \rrbracket \leq \llbracket G(\underline{T}) \rrbracket \sigma_\Gamma = G(\llbracket \underline{T} \rrbracket \sigma_\Gamma)$. Since $\llbracket \psi \rrbracket$ is prime and G is multilinear there exists a vector of primes $\underline{p}$ such that $p_i \leq \llbracket T_i \rrbracket \sigma_\Gamma$ and $\llbracket \psi \rrbracket \leq G(\underline{p})$. Let p_i be denoted by ϕ_i. Then $\llbracket \psi \rrbracket \leq G(\llbracket \phi_1 \rrbracket, \ldots, \llbracket \phi_k \rrbracket) = \llbracket G(\underline{\phi}) \rrbracket$. By the completeness of $\mathcal{L}$ we have $\mathcal{L} \vdash G(\underline{\phi}) \leq \psi$ and since). Also $\llbracket$ *and since*

If ρ is a closed substitution we write $\rho \models^{\mathcal{O}} \Gamma$ if for every $X \in \mathcal{X}$ $\rho(X$

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 **P** is adequate for

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 51, Lecture Notes in Computer Science, pages 169–201, 1987.

[Bar84] Henk Barendregt. *The Lambda Calculus.* North-Holland, 1984. Studies in logic 10 .

[BCDC8] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter model and the completeness of type assignment. *J. of Symbolic Logic*, 48:9 1–940, 198 .

[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 51.

[Bou90a] G. Boudol. Flow eventfor

[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 44 .

[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