Relationally Staged Computations in Calculi of Mobile Processes

Neil Ghani, Kidane Yemane, and Björn Victor

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2004-010
March 2004
ISSN 1404-3203

Relationally Staged Computations in Calculi of Mobile Processes

Neil Ghani[*] Kidane Yemane[†] Björn Victor[†]

February 20, 2004

Abstract

We apply the recently developed techniques of higher order abstract syntax and functorial operational semantics to give a compositional and fully abstract semantics for the π-calculus equipped with open bisimulation. The key novelty in our work is the realisation that the sophistication of open bisimulation requires us to move from the usual semantic domain of presheaves over subcategories of **Set** to presheaves over subcategories of **Rel**. This extra structure is crucial in controlling the renaming of extruded names and in providing a variety of different dynamic allocation operators to model the different binders of the π-calculus.

1 Introduction

The π-calculus was introduced by Milner, Parrow and Walker [MPW92] as an attempt to extend previous paradigms for distributed and concurrent computation with structure to reflect the locality of information. Thus, in addition to the usual combinators of process algebra such as prefix, parallel combination, nondeterministic choice, input and output, the π-calculus contains processes of the form $(\boldsymbol{\nu} x)P$ which should be thought of as the process P with local information x. Crucially, the operational semantics of the π-calculus allows the topography of this locality information to evolve dynamically so that information which is local at some point may become global later. This phenomenon, known as *extrusion*, is the central innovation of the π-calculus. The π-calculus allows processes to change their connectivity over time. Hence it is called a *calculus of mobile processes*.

One of the key advances of research in process algebra was the concept of bisimulation as a method of proving equivalences of processes. While this concept is fairly simple as introduced in CCS [Mil80], the concept becomes more sophisticated in process algebra with variable binding as a variety of different possibilities exist. The simpler ones, such as *early* bisimulation and *late* bisimulation [MPW92] suffer from not being congruences. *Open* bisimulation [San96] on the other hand is a congruence but is complicated since the bisimulation relation must be closed under not only transitions, but also certain substitutions

[*]Dept. of Mathematics and Computer Science, University of Leicester, United Kingdom.

[†]Dept. of Information Technology, Uppsala University, Sweden. Work supported by European Union project PROFUNDIS, Contract No. IST–2001–33100.

and these substitutions vary from process to process. This makes the overall definition rather more complex and subtle than the definition of early and late bisimulation.

The success of the π-calculus, and the variety of different bisimulations it supports, makes it an ideal case study for some exciting recent developments in program language semantics, in particular for *higher order abstract syntax (HOAS)* [FPT99] and *functorial operational semantics (FOS)* [Tur96]. HOAS aims to extend the highly successful paradigm of initial algebra semantics to languages involving variable binding while FOS seeks a uniform framework in which to reason about the relationship between operational and denotational models. Hence we aim at a treatment of open bisimulation in the π-calculus by using the following strategy:

- Since the π-calculus contains variable binding, we follow the HOAS approach and model the syntax of the π-calculus by the initial algebra of an endofunctor over a category of presheaves.

- The observations one can make of a process are given by a language of prefixes which is modelled by a copointed endofunctor over the same category of presheaves. This endofunctor is called the behaviour functor. Transition systems are the coalgebras for this behaviour functor and bisimulations are given by coalgebraic bisimulations.

- We model the operational semantics as a distributivity of the syntax over the behaviour from which we derive compositionality and full abstraction results.

The contributions of this paper are therefore:

- We give a clean presentation of open bisimulation and derive results like compositionality and full abstraction. This is achieved by using abstract ideas such as *coalgebra*, *naturality* and *indexing* to avoid the various technical side conditions concerning free and bound names which appear in the standard definitions [SW01].

- The sophistication of open bisimulation arises from its closure under varying classes of renamings. The novelty of our approach is that we capture this feature by moving from presheaves over a subcategory of **Set** (as in [FT01]) to presheaves over a subcategory of **Rel** which we call relational presheaves. In this paper **Rel** is taken to be the category with relations as objects and monotone functions as morphisms. These relational presheaves offer more dynamic allocation operators to model the binders in the π-calculus than are usually available.

- As we comment on the conclusion, this extra power of relational presheaves seems exactly what is needed to treat other sophisticated bisimulations such as hyperbisimulation in the Fusion calculus.

Related Work: This program of research has been applied to the π-calculus in [FT01] but only to *early* and *late* bisimulations. However, the sophistication of open bisimulation means that these methods could not be applied — our solution of using relational presheaves seems to be exactly what is required in

$\alpha ::=$	$a(x)$	(Input)	$P ::=$	$\mathbf{0}$	(Inaction)
	$\overline{a}x$	(Output)		$\alpha \,.\, Q$	(Prefix)
	$\overline{a}(x)$	(Bound Output)		$Q + R$	(Summation)
	τ	(Silent)		$Q \mid R$	(Composition)
				$(\boldsymbol{\nu} x)Q$	(Restriction)
				$[x = y]Q$	(Match)
				$!\, P$	(Replication)

Figure 1: Syntax of π-calculus

terms of providing appropriate extra structure. Open bisimulation has been written about extensively in the concurrency literature [SW01, PS01]. Interestingly, in these papers, some constructions are indexed by relations as proposed here while others are not. Thus our idea of building our semantics around relational presheaves fits in with, and extends, a trend in the mainstream treatment of open bisimulation. But the fact that one then sees connections with other bisimulations such as hyperbisimulation makes a strong case for moving to our more abstract setting.

This paper will appeal to those coalgebraists who are interested in HOAS and FOS as, in order for this theory to fully mature, more complex examples need to be treated. This is precisely what we have done. For example, we wonder whether the relational presheaves we consider here fit into Power's axiomatics for FOS [Pow03b]. In addition, two of the present authors work in the area of concurrency and came to this subject as they were interested in the possibility of more abstract treatment of their subject. Given the proliferation of different process algebras and the numerous associated bisimulations, this is certainly a growing area of interest and we expect other concurrency theorists to be interested in this work. In summary, there are both good theoretical and practical reasons for the development of open bisimulation within the higher order functorial operational semantics framework that we have provided.

The paper is structured as follows: We provide the concrete syntax and semantics of π-calculus in section 2 followed by abstract syntax of the π-calculus in section 3. Section 4 contains the associated behaviour functor and the key results on the compositional and fully abstract semantics of the π-calculus. We finish in section 5 with some conclusions and ideas for further work.

2 The π-calculus

In this section we recall the syntax and semantics of the π-calculus and the definition of open bisimulation equivalence. As this is standard material, we refer the reader to standard texts for some of the technical details, eg [SW01].

We assume an infinite set of names $\mathcal{N}$ ranged over by lower case $a, b, \ldots x, y, \ldots$. These represent the communication channels and the values sent and received.

Definition 1 (The π-calculus) The set of raw π-calculus processes, ranged over by P, Q, R etc, and prefixes α are defined inductively by the rules of Figure 1.

$$\text{PREF}\ \frac{-}{\alpha\,.\,P \xrightarrow{\alpha} P} \qquad \text{RES}\ \frac{P \xrightarrow{\alpha} P'}{(\boldsymbol{\nu} z)P \xrightarrow{\alpha} (\boldsymbol{\nu} z)P'} \text{ if z} \notin \text{n}(\alpha)$$

$$\text{SUM}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{PAR}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \text{ if bn}(\alpha) \notin \text{fn(Q)}$$

$$\text{MATCH}\ \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'} \qquad \text{COM}\ \frac{P \xrightarrow{u(x)} P', \quad Q \xrightarrow{\overline{u}y} Q'}{P \mid Q \xrightarrow{\tau} P'\{y/x\} \mid Q'}$$

$$\text{OPEN}\ \frac{P \xrightarrow{\overline{x}y} P'}{(\boldsymbol{\nu} y)P \xrightarrow{\overline{x}(y)} P'} \text{ if } x \neq y \qquad \text{CLOSE}\ \frac{P \xrightarrow{u(y)} P', \quad Q \xrightarrow{\overline{u}(y)} Q'}{P \mid Q \xrightarrow{\tau} (\boldsymbol{\nu} y)(P' \mid Q')}$$

$$\text{REP-PREF}\ \frac{P \xrightarrow{\alpha} P'}{!\,P \xrightarrow{\alpha} P' \mid !\,P} \qquad \text{REP-COM}\ \frac{P \xrightarrow{\overline{u}y} P', \quad P \xrightarrow{u(y)} P''}{!\,P \xrightarrow{\tau} P' \mid P''\{y/x\} \mid !\,P}$$

$$\text{REP-CLOSE}\ \frac{P \xrightarrow{\overline{u}(y)} P', \quad P \xrightarrow{u(y)} P''}{!\,P \xrightarrow{\tau} (\boldsymbol{\nu} y)(P' \mid P'') \mid !\,P}$$

Figure 2: Transition rules for π-calculus

$\mathbf{0}$ stands for the empty process which can do nothing. The process $\alpha\,.\,Q$ can perform the relevant input, output or silent action and then become the process Q; $Q+R$ is the nondeterministic choice of Q and R; $Q \mid R$ is the process Q and R running in parallel; $(\boldsymbol{\nu} x)Q$ is the process Q where x occurs locally and is bound in Q; $[x = y]Q$ is the process Q if the names x and y are equal and otherwise is $\mathbf{0}$. Finally $!\,P$ is the replication operator which can be seen as unlimited copies of P and can be used to encode recursion. The prefixes generate the observable behaviour that processes exhibit as they evolve. Observations vary depending upon the bisimulation one wants to study but for open bisimulation there are four, namely i) $a(x)$ represents the input of an unknown name x on channel a; ii) $\overline{a}x$ represents the output of a global name x on channel a; iii) $\overline{a}(x)$ represents the output of a local name x on channel a; and iv) τ represents a silent internal action. In the actions above, we call a the *subject* of the action, and x the *object*.

The input prefix and the restriction operator are binders (the bound output prefix $\overline{a}(x)\,.\,P$ is short for $(\boldsymbol{\nu} x)\overline{a}x\,.\,P$). Thus there are the obvious notions of free and bound names in a process or prefix, denoted $\text{fn}(P), \text{fn}(\alpha), \text{bn}(P), \text{bn}(\alpha)$; we write $\text{n}(P)$ and $\text{n}(\alpha)$ for the free *and* bound names in a process or prefix. The binders then induce the obvious notion of α-equivalence on processes, and the terms of the π-calculus are the processes quotiented by α-equivalence. Having defined the syntax of the π-calculus, we turn to its operational semantics.

Definition 2 The family of *transitions* $P \xrightarrow{\alpha} Q$ is the least family satisfying the laws in Figure 2, where symmetric rules for SUM, PAR, COM and CLOSE have been omitted.

Open bisimulation, originally defined by Sangiorgi [San96], is finer than the early and late bisimulations [MPW92]; unlike early and late bisimulation, open bisimulation is a congruence and, although its definition may seem more complex at first sight, it has an "efficient" characterization exploited in automated tools [San96, VM94]. Open bisimulation is defined as a set of relations indexed by *distinctions* which are used to express the permanent inequality of names.

Definition 3 A *distinction* is a finite symmetric and irreflexive relation on names. We may write $\{(x,y)\}$ for $\{(x,y),(y,x)\}$ etc, omitting symmetric pairs. A *substitution* is a function on names which has obvious actions on processes and distinctions. A substitution σ *respects* a distinction D if $(a,b) \in D$ implies $\sigma(a) \neq \sigma(b)$.

Distinction relations are used to record the names extruded by bound output; these names were originally bound and hence must be kept different from all other names — the use of distinction relations allows us to consider only those renamings which keep extruded names distinct.

Definition 4 A distinction indexed set $\mathcal{S} = \{S_D\}$ of symmetric process relations is an *open bisimulation* if for each S_D and for each σ which respects D, whenever $(P,Q) \in \mathcal{S}_D$,

1. if $P\sigma \xrightarrow{\overline{a}\,(x)} P'$ with $x \notin \text{fn}(Q)$, then there is a Q' such that $Q\sigma \xrightarrow{\overline{a}\,(x)} Q'$ and $(P',Q') \in S_{D'}$, where $D' = D\sigma \cup \{\{x\} \times \text{fn}(P\sigma, Q\sigma)\}$;

2. if $P\sigma \xrightarrow{\alpha} P'$ with $\text{bn}(\alpha) \notin \text{fn}(Q)$ otherwise, then Q' exists s.t. $Q\sigma \xrightarrow{\alpha} Q'$ and $(P',Q') \in S_D$.

P and Q are *open D-bisimilar*, written $P \sim_D Q$, if there is an open bisimulation $\mathcal{S}$ s.t. $(P,Q) \in S_D$ for $S_D \in \mathcal{S}$.

As a simple motivating example, consider the processes

$$\begin{array}{rclcrcl} P & = & \overline{x}\,.\,\mathbf{0} \mid y\,.\,\mathbf{0} & \qquad & P_0 & = & (\boldsymbol{\nu} x,y)\overline{z}\,x\,.\,\overline{z}\,y\,.\,P \\ Q & = & \overline{x}\,.\,y\,.\,\mathbf{0} + y\,.\,\overline{x}\,.\,\mathbf{0} & & Q_0 & = & (\boldsymbol{\nu} x,y)\overline{z}\,x\,.\,\overline{z}\,y\,.\,Q \end{array}$$

where clearly $P \not\sim_\emptyset Q$, since $P\{x/y\} \xrightarrow{\tau} P'$ but $Q\{x/y\} \stackrel{\tau}{\not\longrightarrow} Q'$. However, $P_0 \sim_\emptyset Q_0$, since there is no possibility of x and y ever being identified. This information is recorded in the distinction index of the open bisimulation relation; after the first two bound outputs $\overline{z}\,(w)$ and $\overline{z}\,(y)$, we only need to verify that $P \sim_{\{(x,y)\}} Q$, thus ruling out all substitutions identifying x and y. For further examples and motivation see e.g. [SW01, San96].

We shall see later how these technical conditions concerning bound and free names etc are subsumed within the HOAS set up to provide a more pleasing definition.

3 Abstract Syntax

The problem with the concrete definition of the syntax of the π-calculus as given in section 1 is that it is presented as a quotient datatype. Reasoning with such quotient datatypes is notoriously difficult as one either has to effectively pick representatives for the equivalence classes or work directly with the equivalence classes themselves. Consequently, it was something of a breakthrough when it was discovered how to present not just the raw terms, but also the α-equivalence classes of terms as free or initial datatypes [FPT99].

The approach may be summarised as follows: rather than defining a set of raw terms and then quotienting them by α-equivalence, one defines for each set of free variables, the set of terms definable in that context. These contexts are also called *stages*. Thus for each context or stage we have a set of terms. Allowing renaming of variables in contexts means contexts form a category $\mathcal{C}$ and then terms should be given by a functor $\mathcal{C} \to \mathbf{Set}$. Typically $\mathcal{C}$ has been chosen to be either the category of finite sets and all functions [FT01] the category of finite sets and injective functions [FT01]. Our central insight and the novelty of our approach is that to tackle more sophisticated situations such as open bisimulation in the π-calculus and hyperbisimulation in the Fusion calculus, more structure is required of the stages and in particular that stages form subcategories of **Rel**.

Open bisimulation is closed not just under transitions but also renamings which must be injective on the names which have been extruded. Thus each stage should not consist only of a set of free variables but should include a relation which says when names cannot be renamed to be the same. Thus our presheaves will be over certain subcategories of **Rel** and not **Set**. For open bisimulation, this subcategory is the category of distinction relations which we now define.

3.1 The Category of Distinction Relations

To interpret open bisimulation we use stages which consist of distinction relations. Thus π-calculus-terms will form a presheaf $\mathcal{D} \to \mathbf{Set}$ where $\mathcal{D}$ is the category of distinction relations. The category $\mathcal{D}$ is defined as follows.

Definition 5 (The category $\mathcal{D}$) The category $\mathcal{D}$ of distinctions relations is the full subcategory of **Rel** whose objects are distinction relations, i.e., relations which are irreflexive, symmetric and have a finite carrier set.

Thus morphisms between distinction relations (n, d_n) and (m, d_m) are functions $f : n \to m$ which preserve the distinction relation. Intuitively, a distinction relation is a set and a relation such that related elements are thought of as definitely distinct which must never be renamed to be the same. As we argued earlier, finding a mathematical formalism to ensure that extruded names are renamed injectively while other names may be renamed non-injectively is the key to understanding open bisimulation.

As mentioned above, the main insight here is that we work in the presheaf category $\mathbf{Set}^{\mathcal{D}}$. Much of the required structure of this presheaf category is inherited from $\mathcal{D}$ as we now describe.

Lemma 6 (Structure of $\mathcal{D}$) *The category $\mathcal{D}$ has three distinguished dynamic allocation functors* $\mathsf{Id}, \delta^+, \delta^- : \mathcal{D} \to \mathcal{D}$

Proof of Lemma 6: The dynamic allocation operator Id is the identity, while δ^- is simply coproduct with the distinction relation on the one element set. The action of $\delta^+ : \mathcal{D} \to \mathcal{D}$ on objects is given by $\delta(n, d_n) = (n+1, d_{n+1})$ where d_{n+1} is the symmetric closure of

$$d_n \cup \{(*, i) | i \in n\}$$

The action on morphisms is as expected while functoriality is a simple calculation. □

Thus both δ^- and δ^+ both add an extra element to the carrier of a distinction relation to represent the bound variable. However, δ^+ asks that, in addition, this new element is made distinct from the other elements. The functor δ^+ will be used for the binding associated with restriction to ensure that the extruded name cannot be renamed to other name while the δ^- functor is used for bound input where no such restrictions are necessary. The superscripts $+, -$ are designed to convey the idea that while both δ^+ and δ^- both add in an extra element, δ^+ adds in *extra* distinctions. The presence of more than one dynamic allocation operator is a direct consequence of our move to relational presheaves as a semantic domain. We use the identity as a dynamic allocation operator solely to avoid case analysis later on.

The structure exhibited in Lemma 6 lifts to structure of the category of $\mathcal{D}$-presheaves as follows. Note that the restriction to finite distinction relations means that there are no size problems when talking about the category of $\mathcal{D}$-presheaves. Rather than invent new symbols for the lifted structure, we shall use the same symbols but ensure the reader has enough information to deduce which category we are working in.

Lemma 7 (Structure of $\mathbf{Set}^{\mathcal{D}}$) *The category $\mathbf{Set}^{\mathcal{D}}$ has products, coproducts, a presheaf of names, three dynamic allocation functors $\mathsf{Id}, \delta^-, \delta^+$ and a finite powerset functor defined as follows:*

- *The presheaf of names $N \in \mathbf{Set}^{\mathcal{D}}$ with action $N(n, d_n) = n$.*
- *As with all limits and colimits in functor categories, products and coproducts are computed pointwise. Thus $(P \times Q)(n, d_n) = P(n, d_n) \times Q(n, d_n)$ and similarly for coproducts.*
- *Each dynamic allocation functor $\kappa \in \{\mathsf{Id}, \delta^+, \delta^-\}$ on $\mathcal{D}$ defines the dynamic allocation operator $\_ \circ \kappa : \mathbf{Set}^{\mathcal{D}} \to \mathbf{Set}^{\mathcal{D}}$.*
- *If $\mathcal{P}_f$ is the finite powerset functor on $\mathbf{Set}$, then $\mathcal{P}_f \circ \_ : \mathbf{Set}^{\mathcal{D}} \to \mathbf{Set}^{\mathcal{D}}$ defines the finite power operator on $\mathcal{D}$-presheaves.*

We can now give an initial algebra semantics for the π-calculus based upon distinction relations.

3.2 π-calculus Syntax as an Initial Algebra

The category of presheafs $\mathbf{Set}^{\mathcal{D}}$ provides a suitable universe of types to model syntax and semantics of π-calculus with an open interpretation. The syntax of π-calculus given in Definition 1 is captured upto α-equivalence by an endofunctor $\Sigma : \mathbf{Set}^{\mathcal{D}} \to \mathbf{Set}^{\mathcal{D}}$ defined in Figure 3.

The functor Σ is constructed as expected as the sum of functors for each specific syntactic category of the language. For example, the **0** process always adds one process, while free output prefixed processes consist of two names and a process already constructed. Bound output on the other hand consists of the name of the channel on which the previous local name was to be transmitted, and the process with the extruded name declared to be distinct from all other

$$
\begin{array}{rcll}
\Sigma\, X & = & K_1 & \text{(Inaction)} \\
 & + & N \times N \times X & \text{(Output Prefix)} \\
 & + & N \times \delta^+ X & \text{(Bound Output)} \\
 & + & N \times \delta^- X & \text{(Input Prefix)} \\
 & + & X \times X & \text{(Summation)} \\
 & + & X \times X & \text{(Composition)} \\
 & + & \delta^+ X & \text{(Restriction)} \\
 & + & N \times N \times X & \text{(Match)} \\
 & + & X & \text{(Replication)}
\end{array}
$$

Figure 3: Endofunctor modeling π-calculus-terms

names by the use of δ^+. Compare this situation with the input prefix where the name expected along the input channel is not extruded and hence not forced to be distinct from other names by using δ^-.

The syntax of the π-calculus is the initial Σ-algebra which we write Σ_π. That Σ has an initial algebra follows from the fact that i) $\mathbf{Set}^{\mathcal{D}}$ has ω-colimits and an initial object inherited from **Set**; and ii) sums and products preserve filtered colimits while for any functor F, $\_ \circ F$ always preserves filtered colimits. That $\Sigma_\pi(X)$ is the set of π-calculus terms upto α-equivalence with free names from X is easily seen via the same kind of argument as in [FT01] to establish the correctness of the higher order abstract syntax approach to the untyped λ-calculus. Notice that since Σ_π is a $\mathcal{D}$-presheaf, we can also rename π-calculus terms. However, exactly which names can be equated by such renamings is controlled by the stage or distinction relation where the process lives. Recall that achieving this is the key step to tackling open bisimulation.

4 Labelled Transition Systems

Having given an initial algebra semantics for the syntax of the π-calculus, we turn to a treatment of the open transition relation and hence open bisimulation. Our approach is to define a behaviour functor such that an open transition system is simply a coalgebra for the behaviour functor and open bisimulation is then a coalgebriac bisimulation. Of course, we wish to use the advantages of HOAS to work upto α-equivalence and so the first step is to replace the prefixes of Definition 1 which contain bound variables, with versions upto α-equivalence. Thus we define the set $Act = \{a(), \overline{a}(), \overline{a}b, \tau\}$ and note that if $\sigma : \mathcal{N} \to \mathcal{N}$ is any function over names, then there is the obvious action on the set Act.

Next, we use these actions to define the notion of transition system. In the HOAS framework processes are given in specific stages and hence the nodes of the transition system must be pairs of stages and processes which inhabit them.

Definition 8 ($\mathcal{D}$-transition systems) A $\mathcal{D}$-transition system consists of a presheaf $X : \mathbf{Set}^{\mathcal{D}}$ and a graph such that

- The nodes are labelled by pairs (p, d) where $p \in X(d)$

- The edges are labelled by elements of the set Act such that if
  - $(p, d) \xrightarrow{a()} (p', d')$ then $d' = \delta^- d$ and $a \in d$
  - $(p, d) \xrightarrow{\overline{a}()} (p'd')$ then $d' = \delta^+ d$ and $a \in d$
  - $(p, d) \xrightarrow{\overline{a}b} (p'd')$ then $d' = d$ and $a, b \in d$
  - $(p, d) \xrightarrow{\tau} (p', d')$ then $d' = d$
- If $(p, d) \xrightarrow{\alpha} (p', \kappa d)$ where $\kappa \in \{\mathsf{Id}, \delta^-, \delta^+\}$ and $\sigma : d \to d'$ then $(p\sigma, d') \xrightarrow{\alpha\sigma} (p'\sigma, \kappa d')$.

The essence of the FOS approach is to represent transition systems such as those in Definition 8 as coalgebras of a behaviour functor. We define the behaviour functor as follows:

Definition 9 (Behaviour Functor) The functor $\mathcal{B} : \mathbf{Set}^{\mathcal{D}} \to \mathbf{Set}^{\mathcal{D}}$ is defined as follows

$$\mathcal{B}P = \mathcal{P}_f(P + N \times N \times P + N \times \delta^+ P + N \times \delta^- P)$$

To understand the above definition, think of $\mathcal{B}P$ as the possible evolution of a presheaf of processes P. Non-determinism means there are many possible evolutions and hence the presence of the power type functor $\mathcal{P}_f \circ \_$. Each of these possible evolutions consists of the observable action and the resulting process. The silent and free output actions are the first two possibilities — notice that the resulting processes live in the same stage. For bound output, we observe the channel along which the output is sent and the resulting process has the name extruded and hence the stage is extended by the extruded name which must be kept distinct from all others. Finally, for bound input much the same is true except that in the resulting process the fresh name need not be distinct from the other names.

Lemma 10 *$\mathcal{D}$-transition systems are in one-to-one correspondence with $\mathcal{B}$-coalgebras.*

Proof of Lemma 10: The structure map of the coalgebra gives us exactly the transition relation while the naturality of the coalgebra structure map corresponds exactly to the closure condition of open transitions under distinction preserving renamings. □

A $\mathcal{B}$-coalgebraic bisimulation is defined as follows:

Definition 11 (Coalgebraic Bisimulation) A $\mathcal{B}$-bisimulation over a $\mathcal{B}$-coalgebra X is given by a relation R over X such that there is a $\mathcal{B}$-coalgebra structure on R making the following diagrams commute.

$$\begin{tikzcd} X \arrow[d] & R \arrow[l] \arrow[r] \arrow[d, dotted, "\exists"'] & X \arrow[d] \\ BX & BR \arrow[l] \arrow[r] & BX \end{tikzcd}$$

Lemma 12 *The following data are equivalent:*

- *A $\mathcal{B}$-bisimulation over a presheaf $X : \mathcal{D} \longrightarrow$ **Set***
- *A family of symmetric relations*

$$\{R_d \subset X(d) \times X(d)\}_{d \in \mathcal{D}}$$

such that

  - *If $\sigma : d \to d'$ and $p\ R_d\ q$, then $p\sigma\ R_{d'}\ q\sigma$*
  - *$p\ R_d\ q$ implies if*
    - *$p \xrightarrow{\tau} p'$ then there is a q' such that $q \xrightarrow{\tau} q'$ and $p'\ R_d\ q'$*
    - *$p \xrightarrow{a()} p'$ then there is a q' such that $q \xrightarrow{a()} q'$ and $p'\ R_{\delta^- d}\ q'$*
    - *$p \xrightarrow{\overline{a}()} p'$ then there is a q' such that $q \xrightarrow{\overline{a}()} q'$ and $p'\ R_{\delta^+ d}\ q'$.*
    - *$p \xrightarrow{\overline{a}b} p'$ then there is a q' such that $q \xrightarrow{\overline{a}b} q'$ and $p'\ R_d\ q'$.*

Lemma 12 gives an elegant characterisation of $\mathcal{B}$-bisimulations as open bisimulations but is this really the same notion of open bisimulation as found in Definition 4. The conditions relating to bound names have been implicitly implemented by making processes reside in a specific distinction relation and by using the dynamic allocation operators δ^- and δ^+. The only other issue is that Definition 2.4 is not immediately closed under distinction preserving renaming but this is actually a well known property [SW01].

4.1 Categorical rules

In this section we show how the operational rules of π-calculus given in Figure 2 are modelled via natural transformations of a certain form which imply a compositional semantics with full abstraction property provided that the behaviour functor is finitary and preserves weak pullbacks. That such natural transformations give rise to a compositional and fully abstract semantics goes back to the seminal paper [TP97]. More recent work [LPW00, Pow03a] has recast the abstract theory more elegantly in terms of distributivity laws but, since our goal is to establish the existence of such distributivity laws, we stick with the original concrete format. Since the material here is as one would expect, we treat only the following laws for the restriction operator.

$$\text{RES}\quad \frac{P \xrightarrow{ab} P'}{(\boldsymbol{\nu} z)P \xrightarrow{ab} (\boldsymbol{\nu} z)P'}\ \text{if z} \notin \{\text{a, b}\} \qquad \text{OPEN}\quad \frac{P \xrightarrow{\overline{x}y} P'}{(\boldsymbol{\nu} y)P \xrightarrow{\overline{x}(y)} P'}\ \text{if x} \neq \text{y}$$

In general, we seek to model such operational laws as natural transformations which are natural in X

$$\Sigma(X \times \mathcal{B}X) \to \mathcal{B}TX \quad \text{in} \quad \mathbf{Set}^{\mathcal{D}} \tag{1}$$

In the case of the two rules for restriction given above, we derive the correct natural transformation from a natural transformation $\delta^+(X \times \mathcal{B}X) \to \mathcal{B}TX$ is generated by

$$\delta^+ N \times \delta^+ N \times \delta^+ X \xrightarrow{\rho} \mathcal{P}_f(N \times N \times TX + N \times \delta^+ TX)$$

The definition of ρ can be given using the internal language defined in [FMS96].

$$\rho(a,b,q) = \begin{cases} (a'(), q'), & \textit{if } a \textit{ is } old(a') \textit{ and if } b = new(b') \\ (a'b', (\boldsymbol{\nu} z)q'), & \textit{if } a \textit{ is } old(a') \textit{ and } b = old(b') \\ \emptyset & \textit{if } a = new(a') \end{cases}$$

Where *old* and *new* are the injections $D \longrightarrow \delta^+ D$ and $1 \longrightarrow \delta^+ D$.

Naturality can easily be checked.

Results in [TP97] show the semantics associated to the natural transformation modeling the operational rules induces compositional semantics with full abstraction property ie two process with the same semantics are $\mathcal{B}$-bisimilar.

Lemma 13 *Operational rules of Figure 2 modeled by natural transformation of type(1) induces a compositional semantics having full abstraction property.*

Proof of Lemma 13: From [TP97], we need simply show that the behaviour functor is finitary and preserves weak pullbacks. Most of the constructors of the behaviour functor have already been proven finitary while the power type operator $\mathcal{P}_f \circ \_$ is finitary since $\mathcal{P}_f : \mathbf{Set} \longrightarrow \mathbf{Set}$ is finitary. The fact that $\mathcal{B}$ preserves weak pullbacks follows from i) the fact that products and coproducts do; the fact that $\mathcal{P}_f$ does; iii) and that if $K : D \longrightarrow D$ is any functor then $\_ \circ K : \mathbf{Set}^{\mathcal{D}} \longrightarrow \mathbf{Set}^{\mathcal{D}}$ preserves weak pullbacks. This last claim is elegantly proved using left Kan extensions along K to verify the property directly.

□

5 Conclusion

This paper has given a clean presentation of open bisimulation in the π-calculus using the recently developed theories of higher order abstract syntax and functorial operational semantics. Open bisimulation is more sophisticated than early and late bisimulation which had previously been studied in this framework because the extruded names must be recorded and kept distinct from all other names under renaming. We solve this technical challenge in an elegant way by moving from presheaves over a subcategory of **Set** to presheaves over a subcategory of **Rel**, namely the category of distinction relations. The extra structure of such relational presheaves offers a choice of dynamic allocation operators to model the different binding operators which arise in the language. By verifying the preconditions of the functorial operational semantics framework, we then obtain a compositional and fully abstract semantics for open bisimulation in the π-calculus.

There are a number of future directions we wish to take this research. Firstly relational presheaves seem to offer the extra structure required to model sophisticated bisimulations such as open bisimulations. The fusion calculus [PV98] is a variant of the π-calculus where the communication rule

$$\textsc{com}\ \frac{P \xrightarrow{u(x)} P',\ \ Q \xrightarrow{\overline{u}y} Q',}{P \mid Q \xrightarrow{\tau} P'\{y/x\} \mid Q'}$$

is replaced by the communication rule

$$\text{COM}\ \frac{P \xrightarrow{ux} P',\ \ Q \xrightarrow{\overline{u}y} Q'}{P \mid Q \xrightarrow{\{x=y\}} P' \mid Q'}\ |x| = |y|$$

which can be thought of as an explicit substitution and with all the associated benefits of making substitution local etc. In such a calculus processes are defined in a context which consists of not just a set of free names but also an equivalence relation on the names which arise from the communication rule above. This suggests modeling the fusion calculus using presheaves with stages given by finite equivalence relations. We already have sketched the details and had hoped to include this work in this paper but alas time ran out on us.

In the longer term there are a variety of directions we wish to go. On the theoretical side, it would be very interesting to extend these techniques to higher order process calculi and much of the semantic infrastructure already exists and is waiting to be used. We would also like to consider whether these techniques can be extended to Milner's bigraphs [JM03] which is a formalism intended to abstract away from specific process calculi. Further, it would be interesting to test the applicability of Power's recent work on computing with distributivity laws [Pow03a] to this setting.

A more practical direction to take this research is that of HD-automata [MP98]. HD-automata seek to give an operational model of History Dependent calculi by decorating the states of automata with relevant information (free names of a processes) and establishing the correspondence between the local information in different states. These automata can be seen as concrete realisations of the more abstract framework [PS01, FMP02]. By automatically distilling a concrete, minimized representation of a process as a history dependent automata, we can expect applications in model checking and verification along the lines of [FGM+98].

References

[FGM+98] Gianluigi Ferrari, Stefania Gnesi, Ugo Montanari, Marco Pistore, and Gioia Ristori. Verifying mobile processes in the HAL environment. In Alan J. Hu and Moshe Y. Vardi, editors, *Proceedings of CAV '98*, volume 1427 of *LNCS*. Springer, 1998. Tool Poster.

[FMP02] Gianluigi Ferrari, Ugo Montanari, and Marco Pistore. Minimizing transition systems for name passing calculi: A co-algebraic formulation. In Mogens Nielsen and Uffe H. Engberg, editors, *Proceedings of FoSSaCS 2002*, volume 2303 of *LNCS*, pages 129–143. Springer, April 2002.

[FMS96] Marcelo Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the π-calculus (extended abstract). In *Proceedings of LICS '96*, pages 43–54. IEEE, Computer Society Press, July 1996.

[FPT99] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. Abstract syntax and variable binding (extended abstract). In *Proc. 14th LICS Conf.*, pages 193–202. IEEE, Computer Society Press, 1999.

[FT01] Marcelo Fiore and Daniele Turi. Semantics of name and value passing. In *Proc. 16th LICS Conf.*, pages 93–104. IEEE, Computer Society Press, 2001.

[JM03] Ole Hogh Jensen and Robin Milner. Bigraphs and transitions. *SIGPLAN Not.*, 38(1):38–49, 2003.

[LPW00] Marina Lenisa, John Power, and Hiroshi Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In Horst Reichel, editor, *Proceedings 3rd Workshop on Coalgebraic Methods in Computer*

Science, CMCS'00, Berlin, Germany, 25–26 March 2000, volume 33. Elsevier, Amsterdam, 2000.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.

[MP98] Ugo Montanari and Marco Pistore. History dependent automata. Technical report, Computer Science Department, Università di Pisa, 1998. TR-11-98.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, September 1992.

[Pow03a] John Power. Towards a theory of mathematical operational semantics. In H. Peter Gumm, editor, *Electronic Notes in Theoretical Computer Science*, volume 82. Elsevier, 2003.

[Pow03b] John Power. A unified category-theoretic approach to variable binding. Preceedings of MERLIN 2003, Uppsala, Sweden, August 2003.

[PS01] Marco Pistore and Davide Sangiorgi. A partition refinement algorithm for the π-calculus. *Journal of Information and Computation*, 164(2):264–321, January 2001. An extended abstract appeared in *Proceedings of CAV'96*: 38–49.

[PV98] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS '98*, pages 176–185. IEEE, Computer Society Press, July 1998.

[San96] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996. Earlier version published as Report ECS-LFCS-93-270, University of Edinburgh. An extended abstract appeared in the *Proceedings of CONCUR '93*, LNCS 715.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[TP97] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*$^{\text{th}}$ *LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1997.

[Tur96] Daniele Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Free University, Amsterdam, June 1996.

[VM94] Björn Victor and Faron Moller. The Mobility Workbench — a tool for the π-calculus. In David Dill, editor, *Proceedings of CAV '94*, volume 818 of *LNCS*, pages 428–440. Springer, 1994.

Recent technical reports from the Department of Information Technology

2003-056 Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Forward Reachability Analysis of Timed Petri Nets*

2003-057 Erik Berg and Erik Hagersten: *Low-Overhead Spatial and Temporal Data Locality Analysis*

2003-058 Erik Berg and Erik Hagersten: *StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis*

2003-059 Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*

2003-060 Pierre Flener: *Realism in Project-Based Software Engineering Courses: Rewards, Risks, and Recommendations*

2003-061 Lars Ferm and Per Lötstedt: *Space-Time Adaptive Solution of First Order PDEs*

2003-062 Emilio Tuosto, Björn Victor, and Kidane Yemane: *Polyadic History-Dependent Automata for the Fusion Calculus*

2003-063 Michael Baldamus, Joachim Parrow, and Björn Victor: *Spi Calculus Translated to π-Calculus Preserving May-Testing*

2003-064 Arnim Brüger, Bertil Gustafsson, Per Lötstedt, and Jonas Nilsson: *High Order Accurate Solution of the Incompressible Navier-Stokes Equations*

2003-065 Michael Baldamus, Richard Mayr, and Gerardo Schneider: *A Backward/Forward Strategy for Verifying Safety Properties of Infinite-State Systems*

2004-001 Torsten Söderström, Torbjörn Wigren, and Emad Abd-Elrady: *Maximum Likelihood Modeling of Orbits of Nonlinear ODEs*

2004-002 Pablo Giambiagi, Gerardo Schneider, and Frank D. Valencia: *On the Expressiveness of CCS-like Calculi*

2004-003 Johan Elf, Per Lötstedt, and Paul Sjöberg: *Problems of High Dimension in Molecular Biology*

2004-004 Torbjörn Wigren: *Recursive Prediction Error Identification of Nonlinear State Space Models*

2004-005 Håkan Zeffer, Zoran Radovic, Oskar Grenholm, and Erik Hagersten: *Evaluation, Implementation and Performance of Write Permission Caching in the DSZOOM System*

2004-006 Henrik Löf, Markus Nordén, and Sverker Holmgren: *Improving Geographical Locality of Data for Shared Memory Implementations of PDE Solvers*

2004-007 Erik Nordström, Per Gunningberg, and Christian Tschudin: *Comparison of Gateway Forwarding Strategies in Ad hoc Networks*

2004-008 Pär Samuelsson and Bengt Carlsson: *An Integrating Linearization Method for Static Input Nonlinearities*

2004-009 Henrik Johansson and Johan Steensland: *A Characterization of a Hybrid and Dynamic Partitioner for SAMR Applications*

2004-010 Neil Ghani, Kidane Yemane, and Björn Victor: *Relationally Staged Computations in Calculi of Mobile Processes*

UPPSALA
UNIVERSITET

March 2004
ISSN 1404-3203
http://www.it.uu.se/