Presheaf Models for the π-Calculus

Gian Luca Cattani, Ian Stark and Glynn Winskel

BRICS*, Department of Computer Science, University of Aarhus, Denmark

Abstract. Recent work has shown that presheaf categories provide a general model of concurrency, with an inbuilt notion of bisimulation based on open maps. Here it is shown how this approach can also handle systems where the language of actions may change dynamically as a process evolves. The example is the π-calculus, a calculus for 'mobile processes' whose communication topology varies as channels are created and discarded. A denotational semantics is described for the π-calculus within an indexed category of profunctors; the model is fully abstract for bisimilarity, in the sense that bisimulation in the model, obtained from open maps, coincides with the usual bisimulation obtained from the operational semantics of the π-calculus. While attention is concentrated on the 'late' semantics of the π-calculus, it is indicated how the 'early' and other variants can also be captured.

1 Introduction

The gap between domain theory and the theory of concurrency is narrowing. In particular, the π-calculus, which for a long time resisted all but operational semantics, has yielded to a fully abstract denotational semantics; a key idea was to move to domains indexed by a category of name sets (Stark 1996; Fiore, Moggi, and Sangiorgi 1996; Hennessy 1996). Why add yet another model, based this time not on familiar, complete partial orders but instead on presheaf categories?

Several reasons can be given, even at this preliminary stage.

Models for concurrency are best presented as categories where one can take advantage of the universality of constructions (see Winskel and Nielsen (1995)). If domain theory is to meet models for concurrency, it seems that the points of information in a complete partial order need to be replaced by more detailed objects in a category. A problem with the categories of traditional models is that they do not support higher-order constructions. Presheaf categories, on the other hand, not only include important traditional models like synchronisation trees and event structures,[1] but also support function spaces. We see presheaf models as taking us towards a a new domain theory in which presheaf models are analogous to nondeterministic domains (Hennessy and Plotkin 1979).

Another motivation comes in getting a more systematic and algebraic understanding of bisimulation. A traditional way to proceed in giving a theory to

* **B**asic **R**esearch **i**n **C**omputer **S**cience, a centre of the Danish National Research Foundation.

[1] In the sense that these traditional models embed fully, faithfully and densely in particular presheaf models (Joyal, Nielsen, and Winskel 1996).

a process language has been first to endow it with an operational semantics, provide a definition of bisimulation, follow this by the task of verifying that the bisimulation is a congruence (maybe by modifying it a bit), and then establish proof rules. Often the pattern is standard, but sometimes even getting a passable definition of bisimulation can be tricky, as, for example, when higher-order features are involved. An advantage of presenting models for concurrency as categories has been that they then support a general definition of bisimulation based on open maps (Joyal et al. 1996). We can then exploit the universality of various constructions in showing that they preserve bisimulation (Cattani and Winskel 1997). Presheaf categories come along with a concept of open map and bisimulation. They themselves form a category (strictly a bicategory of profunctors), in which the objects are presheaf categories, yielding ways to combine presheaf categories and their bisimulations.

The problem with general definitions is that it's not always so easy to see what their instances amount to. Indeed, a major task of this paper is showing that the bisimulation on processes of the π-calculus obtained from open maps coincides with a traditional definition (following up on earlier work for value-passing processes (Winskel 1996)). However, the presheaf model for the π-calculus contributes more than this. Along the way, the presheaf model casts light on bisimulation for the π-calculus and why operations preserve it; the normal form for processes in the π-calculus (Fiore et al. 1996), which can be read off from the definition of the model (Theorem 5); and suggests smooth translations between variants of the π-calculus (Section 5). We also claim that, compared to the domain model, the "domain equation" of our model is rather simpler, because we seek a category of paths, not processes; the presheaf construction then fills in the necessary nondeterminism. As a result the system is particularly flexible: here we present the full 'late' π-calculus in detail, but we also sketch how the same category holds models of the 'early' π-calculus and other popular variants.

The recent domain models for the π-calculus lie within a functor category $\mathcal{Cpo}^{\mathcal{I}}$ (Fiore et al. 1996; Stark 1996). Here $\mathcal{I}$ is the category of finite sets of names and injections between them, representing the fact that over time new names may be created and old names relabelled to avoid clashes. Hennessy (1996) has followed the same approach in his model for testing equivalences. The key to capturing the π-calculus is that the categorical requirements of functoriality and naturality give uniformity over varying name sets. Most notably $\mathcal{Cpo}^{\mathcal{I}}$ is cartesian closed, and the function space correctly handles the fact that old processes must be prepared to receive new names. This paper makes the same step up in the presheaf approach, to give a model of the π-calculus not in $\mathcal{Prof}$ but in $\mathcal{Prof}^{\mathcal{I}}$.

1.1 The π-calculus

The version of the π-calculus we use is entirely standard. We summarise it only very briefly here: for discussion and further detail see the original papers (Milner, Parrow, and Walker 1992a,b; Milner 1991). Processes have the following syntax

$$P \;::=\; \bar{x}y.P \;\mid\; x(y).P \;\mid\; \nu x\, P \;\mid\; [x{=}y]P \;\mid\; 0 \;\mid\; P+P \;\mid\; P|P \;\mid\; !P$$

with x and y ranging over some infinite supply of *names*. Note that we include the match operator $[x{=}y]P$, unguarded sum and unguarded replication $!P$. This selection is fairly arbitrary: our model copes equally well with mismatch $[x{\neq}y]P$ and processes defined by recursion, guarded or unguarded. Similarly, it makes no difference if we restrict to one of the popular subsets, such as the asynchronous π-calculus (Boudol 1992).

To simplify presentation we identify processes up to a *structural congruence*, the smallest congruence relation satisfying

$$\begin{array}{cccc} [x=x]P \equiv P & !P \equiv P \,|\, !P & \begin{array}{r} x(y).P \equiv x(z).P[z/y] \\ \nu y\, P \equiv \nu z\, P[z/y] \end{array} & \begin{array}{l} z \notin fn(P) \\ z \notin fn(P) \end{array} \\ P + 0 \equiv P & P + Q \equiv Q + P & (P + Q) + R \equiv P + (Q + R) & \\ P \,|\, 0 \equiv P & P \,|\, Q \equiv Q \,|\, P & (P \,|\, Q) \,|\, R \equiv P \,|(Q \,|\, R). & \end{array}$$

Here $P[z/y]$ denotes capture-avoiding substitution — which may of course require in turn the α-conversion of subexpressions. This equivalence is not as aggressive as the structural congruence of, say, Definition 3.1 in (Milner 1992), which allows name restriction $\nu x(-)$ to change its scope. Nevertheless it cuts down the operational rules we shall need, with none at all for matching and replication. All this is to some degree a matter of taste: if we treat process terms as concrete syntax, with no structural identification, the model is still valid. Indeed full abstraction then allows us to read off the fact that α-conversion, commuting '+' and so forth all respect bisimilarity (replacing, for example, the proofs of Theorems 1 to 9 in (Milner et al. 1992b, §3)).

The operational semantics of processes are given by transitions of four kinds: internal or 'silent' action τ, input $x(y)$, free output $\bar{x}y$ and bound output $\bar{x}(y)$. We denote a general transition by α, and define its free and bound names thus:

$$\begin{array}{lll} fn(\tau) = \emptyset & fn(\bar{x}y) = \{x, y\} & fn(x(y)) = fn(\bar{x}(y)) = \{x\} \\ bn(\tau) = \emptyset & bn(\bar{x}y) = \emptyset & bn(x(y)) = bn(\bar{x}(y)) = \{y\}. \end{array}$$

The transitions that a process may perform are given inductively by the rules in Figure 1. This is a *late* semantics, in that input substitution happens in the (COM) rule when communication actually occurs, rather than at (IN). The chief difference between these rules and Table 2 of (Milner et al. 1992b) is that we let structural congruence do some of the work. Thus there are no symmetric forms for the four right-hand rules, and *sometimes processes must be α-converted before they can interact.* Of course the possible transitions derived are exactly the same as with the original definitions.

A symmetric relation $\mathcal{S}$ between processes is a *bisimulation* if for every $(P, Q) \in \mathcal{S}$ the following conditions hold.

- For $\alpha = \tau, \bar{x}y, \bar{x}(y)$, if $P \xrightarrow{\alpha} P'$ then there is Q' such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{S}$.
- If $P \xrightarrow{x(y)} P'$ then there is Q' such that $Q \xrightarrow{x(y)} Q'$ and for any name z, $(P'[z/y], Q'[z/y]) \in \mathcal{S}$.

$$
\begin{array}{llll}
\text{OUT} & \bar{x}y.P \xrightarrow{\bar{x}y} P & \text{SUM} & \dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \\[2ex]
\text{IN} & x(y).P \xrightarrow{x(y)} P & \text{PAR} & \dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad \begin{array}{r} bn(\alpha) \cap fn(Q) \\ = \emptyset \end{array} \\[2ex]
\text{RES} & \dfrac{P \xrightarrow{\alpha} P'}{\nu x\, P \xrightarrow{\alpha} \nu x\, P'} \; x \notin fn(\alpha) & \text{COM} & \dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q'}{P \mid Q \xrightarrow{\tau} P'[z/y] \mid Q'} \\[2ex]
\text{OPEN} & \dfrac{P \xrightarrow{\bar{x}y} P'}{\nu y\, P \xrightarrow{\bar{x}(y)} P'} \quad x \neq y & \text{CLOSE} & \dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} \nu y(P' \mid Q')}
\end{array}
$$

Fig. 1. Transition rules for π-calculus processes

To check this second condition it is only necessary that z ranges over the free names of P and Q, and one fresh name. This relation is *strong*, in that τ-actions must match, and *late*, in that input actions must match before the transmitted value is known. Two processes are (strong, late) *bisimilar* if there is some bisimulation relating them. We write $P \dot{\sim} Q$ and observe that bisimilarity is itself a bisimulation, and contains all others.

Bisimilarity is preserved by all process constructors except for input prefix $x(y).P$. This is because bisimilarity assumes all names are distinct, while the substitution that happens on input can cause names to become identified. One thus defines processes to be *equivalent* $P \sim Q$ if they are bisimilar under all possible name substitutions. Equivalence is then the smallest congruence containing bisimilarity.

1.2 Presheaf Models and Bisimulation

Let P be a small category. The category of *presheaves over* P, often denoted by $\widehat{\mathsf{P}}$ or by $\mathcal{S}et^{\mathsf{P}^{op}}$, is the category whose objects are contravariant functors from P to $\mathcal{S}et$ (the category of sets and functions) and whose arrows are the natural transformations between such functors.

A category of presheaves, $\widehat{\mathsf{P}}$, is accompanied by the *Yoneda embedding*, a functor $y_{\mathsf{P}} : \mathsf{P} \to \widehat{\mathsf{P}}$, which fully and faithfully embeds P in the category of presheaves.

Via the Yoneda embedding we can regard P essentially as a full subcategory of $\widehat{\mathsf{P}}$. In our applications, the category P is to be thought of as consisting of path objects, or computation-path shapes. The Yoneda Lemma, by providing a natural bijection between $\widehat{\mathsf{P}}(y_{\mathsf{P}}(p), X)$ and $X(p)$, justifies the intuition that a presheaf $X : \mathsf{P}^{op} \to \mathcal{S}et$ can be thought of as specifying for a typical path object p the set $X(p)$ of computation paths of shape p. The presheaf X acts on a morphism $m : p \to q$ in P to give a function $X(m)$ saying how q-paths restrict to p-paths. A presheaf being a colimit of path objects can be thought of as a collection of computation paths glued together by identifying subpaths.

Bisimulation on presheaves is derived from notion of open map between presheaves (Joyal and Moerdijk 1994). Recall, a morphism $h : X \to Y$, between presheaves X, Y, is P-*open* iff for all morphisms $m : p \to q$ in P, the square

$$\begin{tikzcd} X(p) \arrow[d, "h_p"'] & X(q) \arrow[l, "Xm"'] \arrow[d, "h_q"] \\ Y(p) & Y(q) \arrow[l, "Ym"'] \end{tikzcd}$$

is a quasi-pullback, *i.e.* whenever $x \in X(p)$ and $y \in Y(q)$ satisfy $h_p(x) = (Ym)(y)$, then there exists $x' \in X(q)$ such that $(Xm)(x') = x$ and $h_q(x') = y$. (This definition of open map, translates via the Yoneda Lemma to an equivalent path-lifting property of h—see (Joyal et al. 1996).)

We say that presheaves X, Y in $\widehat{\mathsf{P}}$ are P-*bisimilar* iff there is a span of surjective open maps between them. This is equivalent to there being a subobject $R \hookrightarrow X \times Y$ such that the compositions with the projections are surjective open.

The category $\widehat{\mathsf{P}}$, over a small category P, is the free colimit completion of P. In more detail, the Yoneda embedding $y_{\mathsf{P}} : \mathsf{P} \to \widehat{\mathsf{P}}$ satisfies the universal property that for any functor $F : \mathsf{P} \to \mathcal{E}$, where $\mathcal{E}$ is a cocomplete category, there is a colimit-preserving functor $Lan_{y_{\mathsf{P}}}(F) : \widehat{\mathsf{P}} \to \mathcal{E}$, the *left Kan extension of F along* y_{P}, unique to within isomorphism, such that $F \cong Lan_{y_{\mathsf{P}}}(F) \circ y_{\mathsf{P}}$. Consequently, if $G : \widehat{\mathsf{P}} \to \mathcal{E}$ is a colimit-preserving functor then, to within isomorphism, G is the left Kan extension $Lan_{y_{\mathsf{P}}}(G \circ y_{\mathsf{P}})$. Recall, in addition, that $Lan_{y_{\mathsf{P}}}(F)$ is left adjoint to the functor taking Y in $\mathcal{E}$ to the presheaf $\mathcal{E}(F(\text{-}), Y)$.

Here we shall only be interested in left Kan extensions where $\mathcal{E}$ is a presheaf category $\widehat{\mathsf{Q}}$. They, and so any colimit-preserving functor between presheaf categories, preserve open maps. Because they preserve colimits they necessarily preserve surjectivity of maps too (and hence bisimulation). Spelt out:

Lemma 1 (Cattani and Winskel 1997). *Let $G : \widehat{\mathsf{P}} \to \widehat{\mathsf{Q}}$ be any colimit-preserving functor between presheaf categories. Then G preserves (surjective) open maps: If h is a (surjective) P-open map in $\widehat{\mathsf{P}}$, then $G(h)$ is a (surjective) Q-open map in $\widehat{\mathsf{Q}}$.*

Notation: If $F : \mathsf{P} \nrightarrow \widehat{\mathsf{Q}}$ is a functor, we will often write $F_!$ for the left Kan extension $Lan_{y_{\mathsf{P}}}(F)$ and $F^\star$ its right adjoint, mentioned above. This notation is reminiscent of a more usual one (that we shall also employ) that given a functor $F : \mathsf{P} \to \mathsf{Q}$ between small categories writes $F_! \dashv F^\star$ for the left Kan extension $Lan_{y_{\mathsf{P}}}(y_{\mathsf{Q}} F)$ and its right adjoint. In this second case a further right adjoint is also present $F^\star \dashv F_\star$, making $F^\star$ into a colimit preserving functor.

Thus we have a variety of general methods to obtain functors $\widehat{\mathsf{P}} \to \widehat{\mathsf{Q}}$, several of them automatically preserving colimits and hence bisimilarity. The following construction gives a relevant example.

A Lifting Construction Given any category P, define $\mathsf{P}_\perp$ as the category isomorphic to P but for a new initial object $\perp$ freely added. Clearly there exists a full embedding $l(ift) : \mathsf{P} \to \mathsf{P}_\perp$. This gives rise to a triple of adjoint functors, $l_! \dashv l^\star \dashv l_\star$

$$\begin{tikzcd} \widehat{\mathsf{P}} \arrow[r, bend left=50, "l_\star"] \arrow[r, bend right=50, "l_!"'] & \widehat{\mathsf{P}_\perp} \arrow[l, "l^\star"'] \end{tikzcd}$$

where for any presheaf $X \in \widehat{\mathsf{P}}$ and path $p \in \mathsf{P}$ we have $l_\star(X)(p) \cong X(p)$ and $l_\star(X)(\perp) \cong \{\star\}$; from which $l^\star(l_\star(X)) \cong X$.

It is not difficult to prove the following proposition analogous to Lemma 2.2 in (Joyal and Moerdijk 1995).

Proposition 2. *$l_\star$ preserves surjective open maps.*

1.3 Profunctors

There are several equivalent ways of presenting profunctors (also called distributors and bimodules, see Borceux (1994)). For us, a *profunctor* $F : \mathsf{P} \nrightarrow \mathsf{Q}$, between small categories P and Q, is a functor $F : \mathsf{P} \to \widehat{\mathsf{Q}}$. The composition $G \circ F$ of profunctors $F : \mathsf{P} \nrightarrow \mathsf{Q}$ and $G : \mathsf{Q} \nrightarrow \mathsf{R}$ is defined to be the composition of functors

$$(Lan_{y_\mathsf{Q}}(G)) \circ F \ ,$$

which is only defined to within isomorphism. Thus, we obtain a *bicategory*[2] of profunctors $\mathcal{P}rof$.

It is helpful to view $\mathcal{P}rof$ as a category of "nondeterministic domains" (Hennessy and Plotkin 1979; Hennessy 1996), analogous to those obtained as the Kleisli category of a powerdomain monad. We exhibit an adjunction to back up this claim. Write ω-$\mathcal{A}cc$ for the category of finitely accessible categories, the category analogue of algebraic cpo's; morphisms are functors preserving filtered colimits. The functor

$$(-)^0 : \omega\text{-}\mathcal{A}cc \to \mathcal{P}rof$$

takes a finitely accessible category $\mathcal{C}$ to its "basis" $\mathcal{C}^0$, a choice of skeletal subcategory of its finitely presentable elements; a filtered-colimit preserving functor $H : \mathcal{B} \to \mathcal{C}$ is sent to the profunctor $H^0 : \mathcal{B}^0 \nrightarrow \mathcal{C}^0$ such that $H^0(B) = \mathcal{C}(-, H(B)) : (\mathcal{C}^0)^{op} \to \mathcal{S}et$. The functor

$$\widehat{(-)} : \mathcal{P}rof \to \omega\text{-}\mathcal{A}cc$$

takes a profunctor $F : \mathsf{P} \nrightarrow \mathsf{Q}$ to a choice of left Kan extension $Lan_{y_\mathsf{P}}(F)$. There is an equivalence of categories

$$\omega\text{-}\mathcal{A}cc(\mathcal{C}, \widehat{\mathsf{P}}) \simeq \mathcal{P}rof(\mathcal{C}^0, \mathsf{P}) \ ,$$

[2] However, henceforth we won't be pernickety in our category-theoretic terminology, and use terms from traditional category theory, even when constructions, strictly speaking, take place in a bicategory setting.

given by restriction to finitely presentable objects, expressing an adjunction between ω-$\mathcal{A}cc$ and $\mathcal{P}rof$:

$$\begin{tikzcd} \omega\text{-}\mathcal{A}cc \arrow[r, shift left=1.5, "(-)^0"] \arrow[r, phantom, "\perp"] & \mathcal{P}rof \arrow[l, shift left=1.5, "\widehat{(-)}"] \end{tikzcd}$$

With bicategorical quibbles, there is good reason to call $\mathcal{P}rof$ the Kleisli category of the monad $\widehat{(-)}^0$, and to think of the monad as analogous to powerdomains in that it introduces nondeterminism.

Turning the monad around we obtain a comonad $!\mathsf{P} = (\widehat{\mathsf{P}})^0$, amounting to the finite-colimit completion of P, whose co-Kleisli category consists of presheaf categories with filtered-colimit-preserving functors as morphisms. In addition, the bicategory $\mathcal{P}rof$ is rich in constructions which enable us to handle higher-order processes as presheaves. Many of these constructions are associated with $\mathcal{P}rof$ being a model of classical linear logic. The tensor $\otimes$ is of particular interest: on objects $\mathsf{P} \otimes \mathsf{Q}$ is the product of categories $\mathsf{P} \times \mathsf{Q}$; while on morphisms if $F : \mathsf{P} \nrightarrow \mathsf{P}'$ and $G : \mathsf{Q} \nrightarrow \mathsf{Q}'$ are profunctors then $F \otimes G : \mathsf{P} \otimes \mathsf{Q} \nrightarrow \mathsf{P}' \otimes \mathsf{Q}'$ acts according to:

$$((F \otimes G)(p, q))(p', q') \;=\; ((F(p))(p')) \times ((G(q))(q')) \ ,$$

where $\times$ is the product functor in $\mathcal{S}et$. The unit of $\otimes$ is $\mathbf{1}$, the category with a single object and morphism. Defining the linear function space $\mathsf{P} \nrightarrow \mathsf{Q}$ to be the product of categories $\mathsf{P}^{op} \times \mathsf{Q}$, we see the natural bijection

$$\mathcal{P}rof(\mathsf{P}, [\mathsf{Q} \nrightarrow \mathsf{R}]) \;\cong\; \mathcal{P}rof(\mathsf{P} \otimes \mathsf{Q}, \mathsf{R}) \ .$$

It is easy to see that presheaves over $\mathsf{P} \nrightarrow \mathsf{Q}$ correspond to profunctors, and so, to within isomorphism, to colimit-preserving functors from $\widehat{\mathsf{P}}$ to $\widehat{\mathsf{Q}}$. Filtered-colimit-preserving functors are represented, with the help of the exponential !, as presheaves over $!\mathsf{P} \nrightarrow \mathsf{Q}$. Products (&) and coproducts $(+, \Sigma)$ in $\mathcal{P}rof$ coincide on objects, where both are given by coproduct of categories; similarly $\top$ and 0 are both the empty category. Linear involution $\mathsf{P}^{\perp}$ is isomorphic to $\mathsf{P} \nrightarrow \mathbf{1}$ and so to P^{op}. $\mathcal{P}rof$ is compact-closed: par ($\parr$) coincides with tensor ($\otimes$), and $\perp$ with $\mathbf{1}$.

A Diagonal Later, in our treatment of replication in the π-calculus, we shall use a diagonal map, which we now construct. Let P and Q be two small categories. Take $w_{\mathsf{P}} : \mathsf{P} \times \mathsf{Q} \to \mathsf{P}$ and $w_{\mathsf{Q}} : \mathsf{P} \times \mathsf{Q} \to \mathsf{Q}$ to be the two projection functors. By composing with the Yoneda embedding they can be promoted to the status of maps in $\mathcal{P}rof$ and therefore give rise to a universal profunctor $w_{\mathsf{P},\mathsf{Q}} = \langle w_{\mathsf{P}}, w_{\mathsf{Q}} \rangle : \mathsf{P} \otimes \mathsf{Q} \nrightarrow \mathsf{P}\&\mathsf{Q}$.[3] This induces an adjoint pair between the associated presheaf categories:

$$\begin{tikzcd} \widehat{\mathsf{P} \times \mathsf{Q}} \arrow[r, shift left=1.5, "w_!"] \arrow[r, phantom, "\perp"] & \widehat{\mathsf{P}} \times \widehat{\mathsf{Q}} \arrow[l, shift left=1.5, "w^*"] \end{tikzcd} \ .$$

[3] Given the "linear logic" structure of $\mathcal{P}rof$, the map w does correspond to a form of *weakening*.

Since $\mathcal{P}rof$ has products we also have that for any small category P a diagonal arrow $\Delta_{\mathsf{P}} : \mathsf{P} \nrightarrow \mathsf{P}\&\mathsf{P}$ is definable. By composing the extension $\Delta_{\mathsf{P},!}$ with $w^{\star}_{\mathsf{P},\mathsf{P}}$, we obtain a functor

$$d_{\mathsf{P}} : \widehat{\mathsf{P}} \to \widehat{\mathsf{P} \times \mathsf{P}} \ .$$

It is not difficult to prove that for any presheaf X over P and for any two objects p_1, p_2 of P:

$$d_{\mathsf{P}} X \langle p_1, p_2 \rangle \;=\; X(p_1) \times X(p_2) \ .$$

Like the lifting of Section 1.2, this is a *structural* arrow, and it makes sense to establish the following preservation property.

Proposition 3. *Let* P *and* Q *be two small categories; then* $w^{\star}_{\mathsf{P},\mathsf{Q}}$ *preserves surjective open maps and, consequently,* d_{P} *does too.*

1.4 The Functor Category $\mathcal{P}rof^{\mathcal{I}}$

As a π-calculus process evolves, the ambient set of channel names may change. To take account of this variability, our path category for the π-calculus will be indexed by $\mathcal{I}$, the category of finite name sets and injective maps between them. Rather than $\mathcal{P}rof$ itself then, we are specifically interested in the functor category $\mathcal{P}rof^{\mathcal{I}}$.

Our first construction is an *object of names* N, the functor $\mathsf{N} : \mathcal{I} \to \mathcal{P}rof$ given by the composition $\mathcal{I} \subseteq \mathcal{S}et \to \mathcal{C}at \to \mathcal{P}rof$, an inclusion followed by two embeddings. This takes a set $s \in \mathcal{I}$ to the corresponding discrete category, which we write also as s.

Several $\mathcal{P}rof$ operations can be extended pointwise to $\mathcal{P}rof^{\mathcal{I}}$: lifting $(-)_{\perp}$, the involution $(-)^{op}$, tensor '$\otimes$', which is the same as par '$\parr$', and product '&', which is the same as coproduct '+'.

We do not know whether or not $\mathcal{P}rof^{\mathcal{I}}$ is monoidal closed with respect to $\otimes$. There is, however, a Yoneda lemma expressing an isomorphism

$$\widehat{\mathsf{A}(s)} \;\cong\; \mathcal{P}rof^{\mathcal{I}}(\mathcal{I}(s,-), \mathsf{A})$$

for s in $\mathcal{I}$ and A in $\mathcal{P}rof^{\mathcal{I}}$. Thus given objects $\mathsf{A}, \mathsf{B} \in \mathcal{P}rof^{\mathcal{I}}$ there is always a candidate for the function space

$$\text{“}\widehat{(\mathsf{A} \nrightarrow \mathsf{B})(s)}\text{”} \;\cong\; \mathcal{P}rof^{\mathcal{I}}(\mathcal{I}(s,-) \otimes \mathsf{A}, \mathsf{B}) \ ,$$

but this is only meaningful if we can exhibit it as an actual presheaf. Conveniently, for the purposes of the π-calculus it is enough to take function spaces $\mathsf{N} \nrightarrow \mathsf{A}$ and it happens that these do exist. On objects they turn out to be given by

$$(\mathsf{N} \nrightarrow \mathsf{A})(s) \;=\; s \times \mathsf{A}(s) + \mathsf{A}(s+1) \ . \qquad (1)$$

A presheaf over this comprises a pair $\langle F, Y \rangle$, where $F : s \nrightarrow \mathsf{A}(s)$ is a profunctor arrow from the discrete category s, and $Y \in \widehat{\mathsf{A}(s+1)}$.

Paths of $(\mathsf{N} \multimap \mathsf{A})(s)$ can be seen, rather loosely, as elements of the graph of a function. Thus a path in the $s \times \mathsf{A}(s)$-component of (1) we write as $(x{\mapsto}p)$ for name $x \in s$ and path $p \in \mathsf{A}(s)$. A path in the $\mathsf{A}(s+1)$-component we write as $(*{\mapsto}p')$ for $p' \in \mathsf{A}(s+1)$. In a similar spirit we can inject a presheaf $X \in \widehat{\mathsf{A}(s)}$ into the left x-component as $(x{\mapsto}X)$, and a presheaf $Y \in \widehat{\mathsf{A}(s+1)}$ into the right component as $(*{\mapsto}Y)$.

Here we have used a convention that we shall follow throughout, that for any set of names s we write '$*$' for the extra element provided in $s+1$, subscripting '$*_s$' when necessary.

This said, we can now describe the action of the profunctor $(\mathsf{N} \multimap \mathsf{A})(i)$ when $i : s \to s'$ is an arrow in $\mathcal{I}$. On the component $s \times \mathsf{A}(s)$

$$(x \mapsto p) \;\mapsto\; (i(x) \mapsto \mathsf{A}(i)(p))\,, \tag{2}$$

while on $\mathsf{A}(s+1)$

$$(*_s \mapsto p') \;\mapsto\; \sum_{y \notin Im(i)} (y \mapsto \mathsf{A}[i,y](p')) + (*_{s'} \mapsto \mathsf{A}(i+1)(p'))\,. \tag{3}$$

Here $[i,y] : s+1 \to s'$ is the injection that extends i over $s+1$ by taking $*_s$ to $y \in (s' \setminus Im(i))$. Note how in this second equation the path $(*_s{\mapsto}p')$ serves as a 'seed' to set paths for all the fresh names of s' not in the image of i.

To handle name creation we use a construction δ on $\mathcal{P}rof^{\mathcal{I}}$, defined by $\delta\mathsf{A} = \mathsf{A}(\_+1)$. This is in fact also a form of function space from N to A, arising from the construction of Day (1970) which lifts the disjoint union '$+$' of $\mathcal{I}$ to a symmetric monoidal closed structure on $\mathcal{P}rof^{\mathcal{I}}$. Approximately speaking, δA comprises functions that will only accept a fresh name as argument. This allows processes to synchronize on the choice of a local name, as required by the (CLOSE) rule of Figure 1. Stark (1996) expands on this a little.

2 The Equation

We derive a suitable π-calculus path object P in $\mathcal{P}rof^{\mathcal{I}}$ from the following equations:

$$\begin{aligned}
\mathsf{P} &\cong \mathsf{P}_\perp + Out + In \\
Out &= (\mathsf{N} \otimes \mathsf{N} \otimes \mathsf{P}_\perp) + (\mathsf{N} \otimes (\delta\mathsf{P})_\perp) && (4)\\
In &= \mathsf{N} \otimes (\mathsf{N} \multimap \mathsf{P})_\perp && (5)
\end{aligned}$$

Unfolding, the four components of P represent silent action, free output, bound output and input respectively. We give the solution to this equation in two stages: first we describe recursively at each set s the corresponding path category $\mathsf{P}(s)$; and then we specify the profunctor arrow that connnects $\mathsf{P}(s)$ to $\mathsf{P}(s')$ for any injective function $i : s \to s'$.

From the descriptions of the constructors δ and $\multimap$, we can think of the family $\mathsf{P}(-)$ as being recursively described by

$$\mathsf{P}(s) \;=\; \mathsf{P}(s)_\perp + s \times s \times \mathsf{P}(s)_\perp + s \times \mathsf{P}(s+1)_\perp + s \times (s \times \mathsf{P}(s) + \mathsf{P}(s+1))_\perp\,.$$

Our minimal $\mathsf{P}(s)$ is thus a poset, in fact a forest of trees. We have four kinds of root: $\tau.$, $x!y.$, $x!*.$ and $x?$ for any $x, y \in s$. Above these in the order relation we find respectively: $\tau.p$, $x!y.p$, $x!*.p'$, $x?(y{\mapsto}p)$ and $x?(*{\mapsto}p')$, where the last two lie above $x?$ and where p is an object of $\mathsf{P}(s)$ while p' is an object of $\mathsf{P}(s+1)$. The arrows of $\mathsf{P}(s)$ are the prefix order, with for example

$$\begin{aligned} x?(y{\mapsto}p) \leq x?(y{\mapsto}\bar{p}) \quad & \text{iff } p \leq \bar{p} \text{ in } \mathsf{P}(s) \\ x?(*{\mapsto}p') \leq x?(*{\mapsto}\bar{p}') \quad & \text{iff } p' \leq \bar{p}' \text{ in } \mathsf{P}(s+1). \end{aligned}$$

Definition 4. We have already used for the objects of $\mathsf{P}(s)$ a notation suggesting the sequence of "atomic" actions that a path represents. In order to describe the arrow part of the functor P, and later on for the semantics of processes, we now give presheaf analogues of the evident prefix operations on paths.

- For α one of τ or $x!y$ we have a prefixing functor $\alpha : \mathsf{P}(s)_\perp \to \mathsf{P}(s)$. This gives an operation on presheaves $\alpha. = \alpha_! \circ l_\star : \widehat{\mathsf{P}(s)} \to \widehat{\mathsf{P}(s)}$ with

$$\alpha.X(p) \;=\; \begin{cases} \{\star\} & \text{if } p = \alpha. \\ X(p') & \text{if } p = \alpha.p' \\ \emptyset & \text{otherwise.} \end{cases}$$

  Similarly with $Y \in \widehat{\mathsf{P}(s+1)}$ we apply $(x!*)_! \circ l_\star$ to obtain $x!*.Y \in \widehat{\mathsf{P}(s)}$.
- Given $X \in \widehat{\mathsf{P}(s)}$ and $x, y \in s$ we define $x?(y{\mapsto}X) \in \widehat{\mathsf{P}(s)}$ by

$$x?(y \mapsto X)\,(p) \;=\; \begin{cases} \{\star\} & \text{if } p = x? \\ X(p') & \text{if } p = x?(y{\mapsto}p') \\ \emptyset & \text{otherwise.} \end{cases}$$

  As above we could also have obtained this from a path map $(x?(y{\mapsto}\_))$.
- Suppose that $\langle F, Y\rangle$ is a presheaf over $\widehat{(\mathsf{N} \nrightarrow \mathsf{P})(s)}$, *i.e.* a profunctor $F : s \nrightarrow \mathsf{P}(s)$ and a presheaf $Y \in \widehat{\mathsf{P}(s+1)}$. We define

$$x?\langle F, X\rangle(p) \;=\; \begin{cases} \{\star\} & \text{if } p = x? \\ F(y)(p') & \text{if } p = x?(y{\mapsto}p') \\ Y(p') & \text{if } p = x?(*{\mapsto}p') \\ \emptyset & \text{otherwise.} \end{cases}$$

  Once again we could instead have derived this from the operations on paths taking $\langle y, p\rangle \in s \times \mathsf{P}(s)$ to $x?(y{\mapsto}p)$ and $p' \in \mathsf{P}(s+1)$ to $x?(*{\mapsto}p')$.

Lemma 1 and Proposition 2 are sufficient to show that all these functors preserve surjective open maps.

Moving on to the morphism part of P, we need a profunctor $\mathsf{P}(i) : \mathsf{P}(s) \nrightarrow \mathsf{P}(s')$ for every injection $i : s \to s'$. We work by induction on the structure of paths in $\mathsf{P}(s)$. In the base cases minimal paths in $\mathsf{P}(s)$ go to the same in $\mathsf{P}(s')$, regarded via Yoneda as presheaves:

$$\mathsf{P}(i)(\tau.) = \tau. \quad \mathsf{P}(i)(x!y.) = i(x)!i(y). \quad \mathsf{P}(i)(x!*_s.) = i(x)!*_{s'}. \quad \mathsf{P}(i)(x?) = i(x)?$$

The inductive steps are:

$$\begin{array}{rr} \mathsf{P}(i)(\tau.p) = \tau.\mathsf{P}(i)(p) & \mathsf{P}(i)(x!y.p) = i(x)!i(y).\mathsf{P}(i)(p) \\ \mathsf{P}(i)(x!*_s.p') = i(x)!*_{s'}.\mathsf{P}(i+1)(p') & \mathsf{P}(i)(x?(y \mapsto p)) = i(x)?(i(y) \mapsto \mathsf{P}(i)(p)) \\ & \mathsf{P}(i)(x?(* \mapsto p')) = i(x)?(\mathsf{N} \nrightarrow \mathsf{P})(i)(p') \;. \end{array}$$

In the last of these we use the non-trivial action of $(\mathsf{N} \nrightarrow \mathsf{P})(i)$ from (3) to 'fill in' input behaviour on receiving names from $(s' \setminus \mathit{Im}(i))$.

2.1 A Decomposition Result

We will now observe that every presheaf $X \in \widehat{\mathsf{P}(s)}$ decomposes into a sum of disjoint components rooted at one of the minimal path objects $\tau., x!y., x!*., x?$ where $x, y \in s$. The decomposition not only allows us to read off a normal form for presheaves, but also leads to a natural notion of transitions for presheaves.

Let m be a minimal object in $\mathsf{P}(s)$ and $X \in \widehat{\mathsf{P}(s)}$. Any $x \in X(m)$ determines a sub-presheaf C of X as follows:

$$C(p) \;=\; \begin{cases} \{\, y \in X(p) \mid X(m,p)(y) = x \,\} & \text{if } m \le p \\ \emptyset & \text{otherwise} \end{cases}$$

for $p \in \mathsf{P}(s)$, and when $p \le q$ define the function $C(p,q) : C(q) \to C(p)$ by

$$C(p,q)(z) = X(p,q)(z) \quad \text{for } z \in C(q)$$

— because X is a contravariant functor it follows that

$$X(m,p)(X(p,q)(z)) \;=\; X(m,q)(z) \;=\; x$$

so that $X(p,q)(z) \in C(p)$. It is easily checked that C is a presheaf and indeed a sub-presheaf of X because its action on morphisms (p,q), when $p \le q$, restricts that of X.

Notation: In this situation, we shall say that C is a *rooted component* of X at x.

Rooted components of X are pairwise disjoint in the sense that if m, m' are minimal objects of $\mathsf{P}(s)$ and C is a rooted component at $x \in X(m)$ and C' is a rooted component at $x' \in C(m')$, then if $C(p) \cap C'(p) \neq \emptyset$ for any $p \in \mathsf{P}(s)$, then $m = m'$ and $x = x'$. Thus, X is isomorphic to a sum of its rooted components:

$$X \;\cong\; \sum_{m} \sum_{x \in X(m)} C_x \tag{6}$$

where m ranges over minimal objects of $\mathsf{P}(s)$ and C_x is the rooted component of X at x.

We analyse further the form of rooted components of $X \in \widehat{\mathsf{P}(s)}$. A rooted component C_i at $i \in X(\tau.)$ is isomorphic to $\tau.X_i$ where $X_i \in \widehat{\mathsf{P}(s)}$ is given by

$$X_i(p) = C_i(\tau.p), \text{ on objects } p \in \mathsf{P}(s), \text{ and}$$
$$X_i(p,q) = C_i(\tau.p, \tau.q) : X_i(q) \to X_i(p), \text{ on morphisms } p \leq q \text{ of } \mathsf{P}(s).$$

We write $X \xrightarrow{\tau} X'$ when there is $i \in X(\tau.)$ such that $X' = X_i$. The assignment $i \mapsto X_i$ is a bijection between the sets $X(\tau.)$ and $\{X' \mid X \xrightarrow{\tau} X'\}$.

A rooted component C_j at $j \in X(x!y.)$, for $x, y \in s$, is isomorphic to $x!y.X_j$, where $X_j \in \widehat{\mathsf{P}(s)}$ is given by

$$X_j(p) = C_j(x!y.p), \text{ on objects } p \in \mathsf{P}(s), \text{ and}$$
$$X_j(p,q) = C_j(x!y.p, x!y.q), \text{ on morphisms } p \leq q \text{ of } \mathsf{P}(s).$$

We write $X \xrightarrow{x!y} X'$ when there is $j \in X(x!y)$ such that $X' = X_j$. The assignment $j \mapsto X_j$ is a bijection between the sets $X(x!y.)$ and $\{X' \mid X \xrightarrow{x!y} X'\}$.

A rooted component C_k at $k \in X(x!*.)$, for $x \in s$, is isomorphic to $x!*.X_k$, where $X_k \in \widehat{\mathsf{P}(s+1)}$ is given by

$$X_k(p') = C_k(x!*.p'), \text{ on objects } p' \in \mathsf{P}(s+1), \text{ and}$$
$$X_k(p',q') = C_k(x!*.p', x!*.q'), \text{ on morphisms } p' \leq q' \text{ of } \mathsf{P}(s+1).$$

We write $X \xrightarrow{x!*} X'$ when there is $k \in X(x!*.)$ such that $X' = X_k$. The assignment $k \mapsto X_k$ is a bijection between the sets $X(x!*.)$ and $\{X' \mid X \xrightarrow{x!*} X'\}$.

Let C_l be a rooted component at $L \in X(x?)$. Define for any $y \in s$, $X_l^y \in \widehat{\mathsf{P}(s)}$ as

$$X_l^y(p) = C_l(x?(y \mapsto p)), \text{ and}$$
$$X_l^y(p,q) = C_l(x?(y \mapsto p), x?(y \mapsto q)) : X_l^y(q) \to X_l^y(p).$$

Define also $X_l^* \in \widehat{\mathsf{P}(s+1)}$ as

$$X_l^*(p') = C_l(x?(* \mapsto p')), \text{ and}$$
$$X_l^*(p',q') = C_l(x?(* \mapsto p'), x?(* \mapsto q')) : X_l^*(q') \to X_l^*(p').$$

Then X_l can be regarded as a pair $\langle \lambda y.X_l^y, X_l^* \rangle$ with $\lambda y.X_l^y : s \nrightarrow \mathsf{P}(s)$. We write $X \xrightarrow{x?} \langle F, Y \rangle$ when there is $l \in X(x?)$ such that $F(y)$ is isomorphic to X_l^y for every $y \in s$ and Y is isomorphic to X_l^*. The assignment $l \mapsto \langle \lambda y.X_l^y, X_l^* \rangle$ is a bijection between the sets $X(x?)$ and $\{\langle F, Y \rangle \mid X \xrightarrow{x?} \langle F, Y \rangle\}$.

This analysis of rooted components transforms (6) into the following result.

Theorem 5 (Decomposition of Presheaves). *Let* $X \in \widehat{\mathsf{P}(s)}$. *Then*

$$X \cong \sum_{i \in X(\tau.)} \tau.X_i + \sum_{x,y \in s} \sum_{j \in X(x!y.)} x!y.X_j + \sum_{x \in s} \sum_{k \in X(x!*.)} x!*.X_k + \sum_{x \in s} \sum_{l \in X(x?)} x?\langle \lambda y.X_l^y, X_l^* \rangle \ .$$

This gives a decomposition of morphisms, which preserves surjective opens.

Proposition 6. *Let* X, Y *be two presheaves over* $\mathsf{P}(s)$ *with* $f : X \to Y$ *a surjective open map beween them. Then the following "restrictions" of* f *are surjective open:*

$$\begin{aligned}
f_i : X_i \to Y_{i'} \quad & \textit{where } i \in X(\tau.) \textit{ and } i' = f_{\tau.}(i) \\
f_j : X_j \to Y_{j'} \quad & \textit{where } j \in X(x!y.) \textit{ and } j' = f_{x!y.}(j) \\
f_k : X_k \to Y_{k'} \quad & \textit{where } k \in X(x!*.) \textit{ and } k' = f_{x!*.}(k) \\
f_l^y : X_l^y \to Y_{l'}^y \quad & \textit{where } l \in X(x?) \textit{ and } l' = f_{x?}(l) \\
f_l^* : X_l^* \to Y_{l'}^* \quad & \textit{where } l \in X(x?) \textit{ and } l' = f_{x?}(l).
\end{aligned}$$

2.2 Indexed Late Bisimilarity for P

The previous section gave a notion of transitions on presheaves; naturally enough, this leads to a form of bisimilarity.

Definition 7. A P-*late bisimulation* is a family $(R_s)_{s \in \mathcal{I}}$ of symmetric binary relations on presheaves in $\widehat{\mathsf{P}(s)}$ such that for any finite name set s and any two presheaves X, Y over $\mathsf{P}(s)$, if $X \; R_s \; Y$ then

$$\begin{aligned}
X \xrightarrow{\tau} X' \;&\Rightarrow\; \exists Y'.\ Y \xrightarrow{\tau} Y' \;\&\; X' \; R_s \; Y' \\
X \xrightarrow{x!y} X' \;&\Rightarrow\; \exists Y'.\ Y \xrightarrow{x!y} Y' \;\&\; X' \; R_s \; Y' \\
X \xrightarrow{x!*} X' \;&\Rightarrow\; \exists Y'.\ Y \xrightarrow{x!*} Y' \;\&\; X' \; R_{s+1} \; Y' \\
X \xrightarrow{x?} \langle F, X' \rangle \;&\Rightarrow\; \exists \langle G, Y' \rangle.\ Y \xrightarrow{x?} \langle G, Y' \rangle \;\&\; X' \; R_{s+1} \; Y' \\
&\qquad \& \; \forall y \in s.\ F(y) \; R_s \; G(y) \ .
\end{aligned}$$

We say that $X, Y \in \widehat{\mathsf{P}(s)}$ are P-*late bisimilar* iff $X \; R_s \; Y$ for some P-late bisimulation $(R_s)_{s \in \mathcal{I}}$.

Lemma 8. P-*late bisimilarity is an equivalence relation.*

Using Proposition 6 we can show that this P-late bisimilarity corresponds exactly to open map bisimilarity:

Lemma 9. *Suppose* X *and* Y *are presheaves over* $\mathsf{P}(s)$. *Then:*

(i) *If* $f : X \to Y$ *is a surjective open map then* X *and* Y *are* P-*late bisimilar.*

(ii) *If* $X \; R_s \; Y$ *for some* P-*late bisimulation* $(R_s)_{s \in \mathcal{I}}$ *then* X *and* Y *are related by a span of surjective open maps.*

Combining these gives:

Proposition 10. *Two presheaves X and Y over $\mathsf{P}(s)$ are P-late bisimilar if and only if they are connected by a span of surjective open maps.*

Moving to a larger set of free names does not affect P-late bisimilarity.

Proposition 11 (Weakening). *If $X, Y \in \widehat{\mathsf{P}(s)}$ are P-late bisimilar then so are $\mathsf{P}(i)_!(X)$ and $\mathsf{P}(i)_!(Y)$ for any injection $i : s \to s'$.*

Moving to smaller name sets is a little more complicated. For any $i : s \to s'$ in $\mathcal{I}$ define $e_i : \mathsf{P}(s) \to \mathsf{P}(s')$ by induction as follows (omitting the trivial base cases):

$$\begin{aligned} e_i(\tau.p) &= \tau.e_i(p) & e_i(x!y.p) &= i(x)!i(y).e_i(p) \\ e_i(x!*_s.p) &= i(x)!*_{s'}.e_{i+1}(p) & e_i(x?(y \mapsto p)) &= i(x)?(i(y) \mapsto e_i(p)) \\ & & e_i(x?(*_s \mapsto p')) &= i(x)?(*_{s'} \mapsto e_{i+1}(p)) \ . \end{aligned}$$

This differs from $\mathsf{P}(i)$ in having a much simpler action on input of unknowns $x?(*{\mapsto}p')$. Even so e_i^*, which by Proposition 1 preserves open maps, turns out to be a left inverse to $\mathsf{P}(i)_!$. This allows us to prove the following result.

Proposition 12 (Strengthening). *For $X, Y \in \widehat{\mathsf{P}(s)}$ and $i : s \to s'$ in $\mathcal{I}$, if $\mathsf{P}(i)_!(X)$ and $\mathsf{P}(i)_!(Y)$ are P-late bisimilar over $\mathsf{P}(s')$ then so are X and Y over $\mathsf{P}(s)$.*

These results suggest that we could have imposed similar uniformity constraints on the family $(R_s)_{s\in\mathcal{I}}$ in Definition 7: we conjecture that without loss of generality we can require that $\mathsf{R} \rightarrowtail\!\!\!\!\!+ (\mathsf{P} \,\&\, \mathsf{P})$ be a cartesian subobject in $\mathcal{P}rof^{\mathcal{I}}$.

3 Constructions

3.1 A Restriction Operator

We define here the operator that will be used to interpret name restriction in π-calculus processes. It arises as a natural family of profunctor arrows indexed by finite sets s and their elements:

$$\nu_{y\in s} : \mathsf{P}(s) \twoheadrightarrow\!\!\!\!\!+ \ \mathsf{P}(s - \{y\}) \ .$$

In particular we can observe that the family $(\nu_{*\in s+1})_s$ define a natural transformation $\nu : \delta(\mathsf{P}) \dot{\to} \mathsf{P}$.

We define the $\nu_{y\in s}$ simultaneously for all s by induction on the structure of the paths. So for each path in $\mathsf{P}(s)$, according to its structure:

$$\begin{aligned} \nu_{y\in s}(\tau.p) &= \tau.\nu_{y\in s}(p) \\ \nu_{y\in s}(x!z.p) &= \begin{cases} x!z.\nu_{y\in s}(p) & \text{if } x, z \neq y \\ x!*_{s-\{y\}}.\mathsf{P}(b_{s,y})(p) & \text{if } x \neq y \text{ and } z = y \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

$$\nu_{y\in s}(x!*_s.p') = \begin{cases} x!*_{s-\{y\}}.\nu_{y\in s+1}(p') & \text{if } x \neq y \\ \emptyset & \text{otherwise} \end{cases}$$

$$\nu_{y\in s}(x?(z\mapsto p)) = \begin{cases} x?(z\mapsto \nu_{y\in s}(p)) & \text{if } x,z \neq y \\ \emptyset & \text{otherwise} \end{cases}$$

$$\nu_{y\in s}(x?(*_s\mapsto p') = \begin{cases} x?(*_{s-\{y\}}\mapsto \nu_{y\in s+1}(p')) & \text{if } x \neq y \\ \emptyset & \text{otherwise} \end{cases}$$

where $b_{s,y} : s \to (s-\{y\})\cup\{*_{s-\{y\}}\}$ is the bijection that renames y to $*_{s-\{y\}}$. The base cases are the evident instances of these for null p: for example $\nu_{y\in s}(\tau.) = \tau$.

The only complication in this definition is the clauses that ensure restriction correctly turns free output into bound output, as summarised in this result:

Lemma 13. *Let X be a presheaf over $\mathsf{P}(s)$ and let $x,y\in s$. If $X \xrightarrow{x!y} X'$, then $\nu_{y\in s,!}(X) \xrightarrow{x!*_{s-\{y\}}} \mathsf{P}(b_{s,y})_!(X')$.*

Observe that if y,z are two different elements of a set s, then

$$\nu_{z\in s-\{y\}}\circ \nu_{y\in s} = \nu_{y\in s-\{z\}}\circ \nu_{z\in s}.$$

This suggests a definition of ν as a contravariant functor from $\mathcal{I}$ to $\mathcal{P}rof$ with $\nu(s) = \mathsf{P}(s)$ and $\nu(i) : \mathsf{P}(s') \nrightarrow \mathsf{P}(s)$ the restriction, in any order, of the elements of $(s' \setminus \mathit{Im}(i))$.

3.2 Parallel Composition

We give an inductive definition for the parallel composition of two presheaves over $\mathsf{P}(s)$ based on the decomposition result of Section 2.1. We then outline how the same definition arises from an interleaving operation on paths, and so deduce that surjective open maps are preserved.

Definition 14. Let X and Y be two presheaves over $\mathsf{P}(s)$ with the respective decompositions indexed by i,j,k,l and i',j',k',l'. Define $X\|_s Y$, inductively as follows, where $i : s\to s+1$ below is the obvious inclusion function.

$$\begin{aligned}
&\sum_{i\in I}\tau.(X_i\|_s Y) + \sum_{x,y\in s}\sum_{j\in J_{x!y}} x!y.(X_j\|_s Y) + \sum_{x\in s}\sum_{k\in K_x} x!*_s.(X_k\|_{s+1}\mathsf{P}(i)_!(Y)) \\
&\qquad + \sum_{x\in s}\sum_{l\in L_x} x?\langle \lambda y.(X_l^y\|_s Y), X_l^*\|_{s+1}\mathsf{P}(i)_!(Y)\rangle \\
&+ \sum_{i'\in I'}\tau.(X\|_s Y_{i'}) + \sum_{x,y\in s}\sum_{j'\in J'_{x!y}} x!y.(X\|_s Y_{j'}) + \sum_{x\in s}\sum_{k'\in K'_x} x!*_s.(\mathsf{P}(i)_!(X)\|_{s+1} Y_{k'}) \\
&\qquad + \sum_{x\in s}\sum_{l'\in L'_x} x?\langle \lambda y.(X\|_s Y_{l'}^y), \mathsf{P}(i)_!(X)\|_{s+1} Y_{l'}^*\rangle \\
&\quad + \sum_{x,y\in s}\sum_{j\in J_{x!y}}\sum_{l'\in L'_x}\tau.(X_j\|_s Y_{l'}^y) + \sum_{x\in s}\sum_{k\in K_x}\sum_{l'\in L'_x}\tau.\nu_{*\in s+1,!}(X_k\|_{s+1}Y_{l'}^*) \\
&\quad + \sum_{x,y\in s}\sum_{j'\in J'_{x!y}}\sum_{l\in L_x}\tau.(X_l^y\|_s Y_{j'}) + \sum_{x\in s}\sum_{k'\in K'_x}\sum_{l\in L_x}\tau.\nu_{*\in s+1,!}(X_l^*\|_{s+1}Y_{k'})
\end{aligned}$$

Much as with the prefixing operations of Definition 4, a more systematic approach begins with the profunctor arrow

$$\|_{s,\bot} : \mathsf{P}(s)_\bot \otimes \mathsf{P}(s)_\bot \mathrel{-\!\!\!+\!\!\!\rightarrow} \mathsf{P}(s)_\bot$$

that interleaves pairs of paths. Judicious use of Kan extension and the lifting maps of Proposition 2 give the parallel composition of presheaves as a bifunctor

$$\|_s : \widehat{\mathsf{P}(s)} \times \widehat{\mathsf{P}(s)} \longrightarrow \widehat{\mathsf{P}(s)}$$

with the following preservation property:

Proposition 15. *Let X, Y, Z, W be presheaves over $\mathsf{P}(s)$. If maps $f : X \to Z$ and $g : Y \to W$ are surjective open, then so is $f\|_s g : X\|_s Y \to Z\|_s W$.*

3.3 Replication

In the same way that the operational semantics of π-calculus replication vanishes into structural congruence, its presheaf interpretation is built entirely from arrows already at hand. Given any small category P, define the "replicated" category P^∞ by solving the following recursive equation[4]

$$\mathsf{P}^\infty \;=\; \mathbf{1} \,\&\, (\mathsf{P} \otimes \mathsf{P}^\infty).$$

This P^∞ is easily calculated to be equivalent to $\sum_{n\in\omega} \mathsf{P}^n$, where $\mathsf{P}^0 = \mathbf{1}$ and $\mathsf{P}^{n+1} = \mathsf{P} \otimes \mathsf{P} \cdots \otimes \mathsf{P}$, $(n+1)$ times. Taking now the diagonal and parallel composition maps at $\mathsf{P}(s)$ we can inductively define replicated versions

$$d^\infty_s : \widehat{\mathsf{P}(s)} \to \widehat{\mathsf{P}^\infty(s)} \quad \text{and} \quad \|^\infty_s : \widehat{\mathsf{P}^\infty(s)} \to \widehat{\mathsf{P}(s)}.$$

Replication itself is simply their composition $!_s = \|^\infty_s \circ d^\infty_s$, and the inductive definition ensures for any presheaf X over $\mathsf{P}(s)$ that $!_s(X) = X \,\|_s\, !_s(X)$.

4 The Interpretation

4.1 Semantics

Following (Stark 1996), we give the interpretation to process terms in two steps. First we associate a process P with free names in s to a presheaf $(\!|P|\!)_s \in |\widehat{\mathsf{P}(s)}|$. Then later, in the full interpretation, we take account of all possible name substitutions by giving a process P with free names s a denotation as a natural transformation:

$$[\![P]\!] : N^{|s|} \stackrel{\cdot}{\longrightarrow} \mathsf{P}\,.$$

[4] Note the relationship beween the way P^∞ is defined and the way the linear logic ! can be defined in presence of infinite products.

Let s be a set of names. For π-calculus processes whose free names lie in s we inductively define:

$$\begin{array}{lll}
(\!|0|\!)_s = \emptyset & (\!|P+Q|\!)_s = (\!|P|\!)_s + (\!|Q|\!)_s & (\!|[x=x]P|\!)_s = (\!|P|\!)_s \\
(\!|\bar{x}y.P|\!)_s = x!y.(\!|P|\!)_s & (\!|P \mid Q|\!)_s = (\!|P|\!)_s \|_s (\!|Q|\!)_s & (\!|[x=y]P|\!)_s = \emptyset \text{ if } (x \neq y) \\
(\!|!P|\!)_s = !_s((\!|P|\!)_s) & (\!|\nu x\, P|\!)_s = \nu_{x \in s+\{x\}}((\!|P|\!)_{s+\{x\}}) &
\end{array}$$

$$(\!|x(y).P|\!)_s = x?\langle F, Y\rangle \text{ where } F(z) = (\!|P[z/y]|\!)_s \text{ for any } z \in s, \\ \text{and } Y = (\!|P[*_s/y]|\!)_{s+1}.$$

Lemma 16. *Let $i : s \to s'$ be an injective function between finite sets, with $\mathbf{x} = \langle x_1, x_2, \ldots, x_{|s|}\rangle$ the names in s. Then for any process P using these names,*

$$\mathsf{P}(i)_!((\!|P|\!)_s) \;\cong\; (\!|P[i(\mathbf{x})/\mathbf{x}]|\!)_{s'} \;.$$

The free names of a process may be bound differently in different contexts. To cope with this, we interpret a process P with $|s|$ free names as a natural transformation $[\![P]\!] : \mathsf{N}^{|s|} \stackrel{\cdot}{\longrightarrow} \mathsf{P}$, where

$$\begin{array}{rl}
[\![P]\!]_{s'} : & \overbrace{s' \times s' \cdots \times s'}^{|s|-\text{times}} \;\longrightarrow\!\!\!\!\to\; \mathsf{P}(s') \\
& \langle a_1, a_2, \ldots, a_{|s|}\rangle \longmapsto (\!|P[\mathbf{a}/\mathbf{x}]|\!)_{s'}
\end{array}$$

Thus the denotation of a process with free names s carries an environment $N^{|s|}$ as a parameter. The proof that this is indeed a natural transformation depends on Lemma 16, that the $(\!|-|\!)$-interpretation respects name substitution.

4.2 Full Abstraction

We can now show our major result, that bisimulation between processes in the π-calculus coincides with that obtained in the model via open maps.

The first two propositions establish a bisimulation between a process P with free names in s and its denotation $(\!|P|\!)_s$.

Proposition 17. *Let P be a process whose free names lie in s. Then*

- $P \xrightarrow{\bar{x}y} Q$ *implies* $(\!|P|\!)_s \xrightarrow{x!y} (\!|Q|\!)_s$
- $P \xrightarrow{\bar{x}(y)} Q$ *implies* $(\!|P|\!)_s \xrightarrow{x!*_s} (\!|Q[*_s/y]|\!)_{s+1}$
- $P \xrightarrow{x(y)} Q$ *implies* $(\!|P|\!)_s \xrightarrow{x?} \langle F, Y\rangle$ *with* $F(z) \cong (\!|Q[z/y]|\!)_s$ *and* $Y \cong (\!|Q[*_s/y]|\!)_{s+1}$
- $P \xrightarrow{\tau} Q$ *implies* $(\!|P|\!)_s \xrightarrow{\tau} (\!|Q|\!)_s$

Proposition 18. *Let P be a process whose free names lie in s. Then*

- $(\!|P|\!)_s \xrightarrow{x!y} X$ *implies* $\exists Q$ *with* $P \xrightarrow{\bar{x}y} Q$ *and* $(\!|Q|\!)_s \cong X$
- $(\!|P|\!)_s \xrightarrow{x!*_s} X$ *implies* $\exists Q, y$ *with* $P \xrightarrow{\bar{x}(y)} Q$ *and* $(\!|Q[*_s/y]|\!)_{s+1} \cong X$

– $(\!|P|\!)_s \xrightarrow{x?} \langle F, Y\rangle$ *implies* $\exists Q, y$ *with* $P \xrightarrow{x(y)} Q$ *and* $(\!|Q[*_s/y]|\!)_{s+1} \cong Y$ *and* $F(z) \cong (\!|Q[z/y]|\!)_s$
– $(\!|P|\!)_s \xrightarrow{\tau} X$ *implies* $\exists Q$ *with* $P \xrightarrow{\tau} Q$ *and* $(\!|Q|\!)_s \cong X$.

Using these results and Proposition 10 we can deduce the following.

Theorem 19. *Let P and Q be two π-calculus processes with free names in s. Then P is late bisimilar to Q if and only if $(\!|P|\!)_s$ and $(\!|Q|\!)_s$ are connected by a span of surjective open maps.*

Suppose now that P is a π-calculus process with free names s_P. Then for any larger set of names s, an injection $i : s_P \to s$ induces a natural transformation $\pi^{s_P,i} : \mathsf{N}^{|s|} \dot{\to} \mathsf{N}^{|s_P|}$ that projects $|s|$-tuples of names to $|s_P|$-tuples. When i is simply an inclusion and no confusion arises we write this as π^{s_P}.

Theorem 20. *Let P and Q be two π-calculus processes with free names s_P and s_Q respectively. Take $s_{P,Q}$ to be the union $s_P \cup s_Q$. Then P is late equivalent (bisimulation congruent) to Q if and only if for any finite set s and any $|s_{P,Q}|$-tuple $\mathbf{a}$ of elements of s, $[\![P]\!]_s \pi_s^{s_P}(\mathbf{a})$ and $[\![Q]\!]_s \pi_s^{s_Q}(\mathbf{a})$ are connected by a span of surjective open maps.*

Note that it is sufficient here to take s to be exactly the free names $s_{P,Q}$ of the two processes. We can also present this result using the 2-categorical setting of our model:

Corollary 21. *Let P and Q be two π-calculus processes with free names s_P and s_Q respectively. Then P is late equivalent to Q if and only if $[\![P]\!] \circ \pi^{s_P}$ and $[\![Q]\!] \circ \pi^{s_Q}$ are connected by a span of modifications whose components are surjective open maps.*

These results show a precise correspondence between operational and denotational notions of process equivalence; notice though that we do not interpret equivalent process by *equal* elements in the model. This could be arranged, by quotienting the categories $\widehat{\mathsf{P}(s)}$ to make every open map invertible, but there seems little reason to do so. In particular this is not likely to be very well-behaved as open maps do not form a calculus of fractions.

5 Other Results and Future Work

The model we have described is just one drawn from a spectrum of name-passing process calculi that can be described within $\mathcal{Prof}^{\mathcal{I}}$. We outline some possibilities.

5.1 Late *vs.* Early

We have given the π-calculus here in its late version, where a process $x(y).P$ carries out input in two stages: it first synchronizes with another process that is prepared to send on channel x; then, later, the transmitted value is substituted for y in the body of P. There is an alternative *early* semantics where these two steps happen together and processes synchronize on (channel,value) pairs. The operational consequences of this choice are discussed in (Milner et al. 1992b, §2.3). There is a corresponding early bisimulation '$\dot\sim_E$' and early equivalence '$\sim_E$', which are both strictly coarser than their late forms.

We can follow these late and early alternatives in our denotational semantics. In presheaf models, synchronization points are marked by lifting $(-)_\perp$ in the equation for the path category. An early version of (5) would be

$$In_E \;=\; N \otimes (N \nrightarrow \mathsf{P}_\perp) \tag{7}$$

where instead of paths $x?$, $x?(y{\mapsto}p)$ and $x?(*{\mapsto}p')$ we now have $x?y$, $x?*$, $x?y.p$ and $x?*.p'$. Solving this new equation in $\mathcal{P}\!\mathit{rof}$ gives an object P_E, and we conjecture that this provides a fully abstract model for the early π-calculus, for suitably adjusted interpretation functions $(\![-]\!)^E$ and $[\![-]\!]^E$.

More directly, this early model seems to arise as a collapse of the late one. A recursively defined morphism of path categories, $k : \mathsf{P}_E \to \mathsf{P}$, induces an arrow in $\mathcal{P}\!\mathit{rof}^{\mathcal{I}}$

$$k^\star : \mathsf{P} \twoheadrightarrow \mathsf{P}_E \tag{8}$$

which we conjecture maps the late semantics for processes into the early one:

$$k^\star((\![P]\!)_s) \;=\; (\![P]\!)^E_s \;\in \mathsf{P}_E(s) \qquad\qquad k^\star \circ [\![P]\!]_s \;=\; [\![P]\!]^E_s \;: N^s \to \mathsf{P}_E$$

for any process P with free names in s.

We can even move the synchronization point for output: clause (4) has a later variant

$$Out_L \;=\; N \otimes (N \otimes \mathsf{P})_\perp \;. \tag{9}$$

It turns out that this makes no difference to process bisimilarity, but it does correspond closely to the presentation style of (Milner 1991). There processes synchronize on channel names alone, $P \xrightarrow{\bar{x}} C$ or $P \xrightarrow{x} F$, becoming *concretions* C (name-process pairs, $N \otimes \mathsf{P}$) and *abstractions* F (name-to-process functions, $N \nrightarrow \mathsf{P}$) respectively. Actual communication is represented by the application of abstractions to concretions $F \bullet C$.

The domain models of the π-calculus in (Fiore et al. 1996; Stark 1996) do not cover the early version, chiefly because rearrangements like equation (7) are harder to express. There the domain equation for processes uses the Plotkin powerdomain to mark synchronization; while our equation for paths uses the much simpler lifting operation.

5.2 Other π-calculi

The development of our model has been purely denotational, with no operational manipulation of processes through expansion laws or the like. As a consequence, there are no *required* operators in the language, and the model remains valid for any subset of the π-calculus. Even so, particular sublanguages may fit simpler equations. For example, the asynchronous π-calculus of Boudol (1992) constrains output to the form $\bar{x}y.0$, suggesting the clause

$$Out_A = N \otimes N \tag{10}$$

to replace (4). The πI-calculus of Sangiorgi (1995) allows only bound output $\bar{x}(y).P$, equivalence to $\nu y(\bar{x}y.P)$ in the original π-calculus. Every communication now passes a fresh name, and we would replace (5) and (4) with

$$In_I = Out_I = N \otimes \delta \mathsf{P} . \tag{11}$$

Moreover the morphism $\mathsf{P}(i)$ now arises from the category map e_i introduced just before Proposition 12, with the restrictions $\nu(i)$ from the end of Section 3.1 being e_i^*, the left inverse, and now also right adjoint, to $\mathsf{P}(i)_!$. This gives some support to Sangiorgi's claim that the πI-calculus is a simpler, more symmetric version of the π-calculus.

These examples show the flexibility of our approach by drawing on the rich categorical structure of $\mathcal{P}rof^{\mathcal{I}}$. As ever in category theory, this also leads us to look at the maps *between* models: we hope to find further morphisms like (8), from 'late' to 'early', that might tie together the wide selection of customized π-calculi proposed in recent years.

We do not have a general function space in $\mathcal{P}rof^{\mathcal{I}}$ so variability of names together with process-passing in systems like CHOCS (Thomsen 1993) or even the full higher-order π-calculus of Sangiorgi (1992) do not yet fit into our framework. Generally, the problem with treating higher-order processes lies not just in writing down and solving plausible "domain" equations, but also in understanding the operational content of the semantics and bisimilarities that arise.

Bibliography

Francis Borceux. *Handbook of Categorical Algebra, vol. 1*, volume 50 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, 1994.

Gérard Boudol. Asynchrony and the π-calculus. Rapport de recherche 1702, INRIA, Sophia Antipolis, 1992.

Gian Luca Cattani and Glynn Winskel. Presheaf models for concurrency. In *Computer Science Logic 1996*, To appear in Lecture Notes in Computer Science. Springer-Verlag, 1997. A preliminary version appeared as BRICS Report RS-96-35.

Brian J. Day. On closed categories of functors. In *Reports of the Midwest Category Seminar IV*, Lecture Notes in Mathematics 137, pages 1–38. Springer-Verlag, 1970.

Marcelo Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the π-calculus. In (LICS 1996).

Matthew Hennessy. A fully abstract denotational semantics for the π-calculus. Technical Report 96:04, School of Cognitive and Computing Sciences, University of Sussex, 1996.

Matthew Hennessy and Gordon Plotkin. Full abstraction for a simple parallel programming language. In *Mathematical Foundations of Computer Science: Proceedings of the 8th International Symposium*, Lecture Notes in Computer Science 74, pages 108–120. Springer-Verlag, 1979.

André Joyal and Ieke Moerdijk. A completeness theorem for open maps. *Annals of Pure and Applied Logic*, 70:51–86, 1994.

André Joyal and Ieke Moerdijk. *Algebraic Set Theory*, volume 220 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 1995.

André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

LICS 1996. *Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, July 27–30, 1996.* IEEE Computer Society Press, 1996.

Robin Milner. The polyadic π-calculus — a tutorial. Technical Report ECS-LFCS-91-180, Laboratory for Foundations of Computer Science, University of Edinburgh, 1991.

Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.

Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I. *Information and Computation*, 100:1–40, 1992a.

Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part II. *Information and Computation*, 100:41–77, 1992b.

Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD Thesis CST-99-93, Department of Computer Science, University of Edinburgh, 1992.

Davide Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. Rapport de recherche 2539, INRIA, Sophia Antipolis, 1995.

Ian Stark. A fully abstract domain model for the π-calculus. In (LICS 1996), pages 36–42.

Bent Thomsen. Plain CHOCS. *Acta Informatica*, 30:1–59, 1993.

Glynn Winskel. A presheaf semantics of value-passing processes. In *CONCUR '96: Proceedings of the 7th International Conference on Concurrency Theory*, Lecture Notes in Computer Science 1119, pages 98–114. Springer-Verlag, 1996. An extended version appears as BRICS Report RS-96-44.

Glynn Winskel and Mogens Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, volume IV. Oxford University Press, 1995.