

Presheaf semantics for HOPLA

Lecture notes
Mikkel Nygaard, BRICS

IT-C, 24th June 2002

1 Introduction

Process calculi like CCS have been motivated and studied operationally, thus from the outset lacking the abstract mathematical treatment provided by a domain theory. Consequently, concurrency has become a rather separate study; in particular, higher-order and functional features as known from sequential programming are most often treated in an ad hoc fashion, if at all.

The study of presheaf models of processes [1, 10] can be seen as an attempt to bring concurrency back within the realm of traditional denotational semantics by providing a domain theory for concurrent computation. In particular, presheaf models come with a built-in notion of bisimulation, derived from open maps [11].

Much of the work so far [2, 4, 5, 6, 7, 8, 17] has concentrated on developing the domain theory itself and on showing how to handle existing models and notions from process calculi within it. Meanwhile, a full operational understanding of presheaf models has still not been obtained. A sensible way to proceed would be to

- exploit the domain theory to define mathematically natural process calculi; and
- approach an operational understanding of presheaf models by investigating the operational semantics of these calculi.

The papers [14, 15] report on joint work with Glynn Winskel along those lines and these notes can be seen as an introduction to that material.

We start with the idea of representing a process as a collection of its possible computation paths, straightforwardly generalised by the categorical notion of a presheaf. We then discuss the basic category theory of presheaves, leading naturally to a

model of linear logic. We consider defining a linear process language based on this model, but find the linearity constraints too restrictive.

In [14] one response to this is discussed: move to a model of affine-linear logic to allow operations to ignore their arguments. The paper goes on to define an expressive affine-linear process language, but unfortunately, its operational semantics is proving difficult. In particular, it has not been extended to higher order; a straightforward attempt with actions of the form $u \mapsto a$ does not seem to work for the affine-linear language, but led us to the operational semantics of HOPLA.

The denotational semantics of HOPLA is obtained by considering a second response to the limitations of linearity: to allow arbitrary copying, which turns the model of linear logic into a cartesian closed category. We'll consider the correspondence between the operational and denotational semantics of HOPLA, and touch upon some unanswered questions concerning bisimulation.

2 Path-based models

Consider processes which, like those of CCS, can perform simple atomic actions, one at a time, among which may be actions of synchronisation. An old idea is to represent the nondeterministic behaviour of such a process as a “collection” of the computation paths it can perform. The trace model [9] and tree model [16] of processes are based on different ideas of what this means. A trace set of a process simply expresses whether or not a finite sequence of actions, a trace, is possible for the process. A tree expresses not only what paths are present but also how paths are subpaths, or restrictions, of others, thus keeping track of nondeterministic branching.

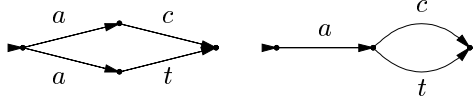


Figure 1: $a.c.\emptyset + a.t.\emptyset$ and $a.(c.\emptyset + t.\emptyset)$

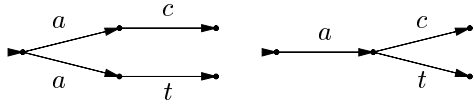


Figure 2: Unfoldings

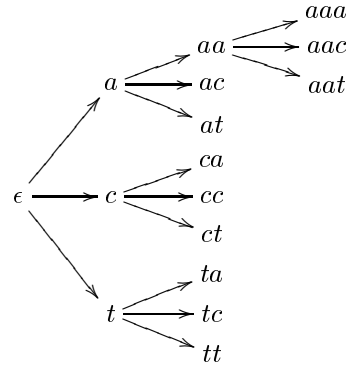


Figure 3: Part of the poset/category A^*

Example 2.1 Consider the two CCS processes of Figure 1. They represent vending machines accepting coins and giving out coffee or tea. Their unfoldings as trees on Figure 2 are clearly different, but as trace sets they are both described by

$$\{\epsilon, a, ac, at\} .$$

As a result, trace sets cannot explain the possible deadlock encountered by a coffee drinker process $\bar{a}.z.\emptyset$ interacting with the version on the left of the figure. \square

3 Presheaves

The data given by the tree representation, what paths are present and how they restrict to smaller paths, is precisely that caught in a presheaf over a category in which the objects are path shapes and the maps express how one path shape can extend to another. In the category of all such presheaves we can view the tree as a *colimit* of its paths—another kind of “collection”.

To illustrate the idea, suppose that actions are drawn from the alphabet $A = \{a, c, t\}$, and consider processes whose computation paths have the shape of strings of actions, so members of A^* . The substring ordering \leq makes A^* a poset, and so a category with an arrow from s to s' precisely when $s \leq s'$, see Figure 3.

A presheaf over A^* is a functor from the opposite category $(A^*)^{\text{op}}$, where all the arrows are reversed, to the category of sets and functions **Set**. When thinking of a presheaf X as representing a process, for a string s , the set $X(s)$ is the set of computation paths of shape s that the process can perform, and,

when $s \leq s'$, the function

$$X(s, s') : X(s') \rightarrow X(s)$$

tells how paths of shape s' restrict to subpaths of shape s . For example, a tree whose branches consists of strings in A^* is easily viewed as a presheaf X over A^* . The set $X(s)$ consists of all branches of shape s , while the function $X(s, s') : X(s') \rightarrow X(s)$ restricts a branch of shape s' to its sub-branch of shape s .

Example 3.1 Let's write $\mathbf{0}$ for the empty set, $\mathbf{1}$ for $\{\mathbf{0}\}$, and $\mathbf{2}$ for $\{\mathbf{0}, \mathbf{1}\}$. The vending machines on the left of Figure 2 is represented by the presheaf

$$X(s) = \begin{cases} \mathbf{2} & \text{if } s = a \\ \mathbf{1} & \text{if } s \in \{\epsilon, ac, at\} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where $X(a, ac)$ maps $\mathbf{0} \in X(ac)$ to $\mathbf{0} \in X(a)$ and $X(a, at)$ maps $\mathbf{0} \in X(at)$ to $\mathbf{1} \in X(a)$. On other morphisms, the action of X is uniquely determined.

Similarly, the presheaf

$$Y(s) = \begin{cases} \mathbf{1} & \text{if } s \in \{\epsilon, a, ac, at\} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

over A^* represents the vending machine on the right of Figure 2. \square

A note on presheaves like Y : Suppose that we replace the category of sets used in the definition of

presheaves by the subcategory $\mathbf{2}$ with the inclusion $\mathbf{0} \hookrightarrow \mathbf{1}$ as the only non-identity arrow. A functor Y from $(A^*)^{\text{op}}$ to $\mathbf{2}$ is the same as a monotonic function from the reverse order $(A^*)^{\text{op}}$ to the order $\mathbf{2}$, so that if $s \leq s'$ then $Y(s') \leq Y(s)$. When thinking of Y as representing a process, $Y(s) = \mathbf{1}$ means that the process can perform a path of shape s while $Y(s) = \mathbf{0}$ means that it can't. If $Y(s') = \mathbf{1}$ and $s \leq s'$, then $Y(s) = \mathbf{1}$. In other words, the functor Y is a characteristic function for a trace set.

So trees and trace sets arise as variants of a common idea, that of representing a process as a generalised characteristic function, in the form of a functor from path shapes to measures of the extent to which the path shapes can be realised by the process.

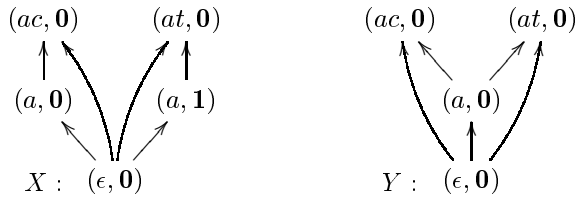
In what follows, we want to broaden computation paths to have more general shapes than sequences of atomic actions, to allow individual actions to have a more complicated structure as they have in HOPLA. So we will consider presheaves over an arbitrary small category \mathbb{P} , that is, functors $X : \mathbb{P}^{\text{op}} \rightarrow \mathbf{Set}$. The category \mathbb{P} is to be thought of as consisting of computation-path shapes where a map $e : p \rightarrow p'$ expresses how the path p is extended to the path p' .

We'll investigate what elementary category theory has to say about this data, thereby explaining a variety of process concepts in terms of "abstract nonsense".

4 Category of elements

Definition 4.1 Consider a presheaf X over \mathbb{P} . The *category of elements* of X , written $\mathcal{E}(X)$, has as objects pairs (p, x) with $p \in \mathbb{P}$ and $x \in X(p)$ and morphisms of the form $e : (p, x) \rightarrow (p', x')$ where $e : p \rightarrow p'$ is an arrow of \mathbb{P} such that $(Xe)x' = x$.

Example 4.2 The categories of elements of the presheaves X and Y of Example 3.1 look as follows (identity arrows are omitted):



Notice that the arrows of these categories may be seen as the reflexive, transitive closure of the transition relations of the trees of Figure 2. \square

So the category of elements can be understood intuitively as the tree or transition system of a process represented by a presheaf. But beware that the system may not have an initial state as it does for X and Y of the example; they are both *rooted* according to

Definition 4.3 A presheaf X over \mathbb{P} is *rooted* when \mathbb{P} has an initial object, \perp , and $X(\perp)$ is a singleton.

A rooted presheaf over A^* corresponds directly to a tree whose branches are in A^* .

In general, a presheaf X over A^* determines a set of such trees, the set of roots given by $X(\epsilon)$. For example, we may consider the diagram above as *one* presheaf Z over A^* , in which $Z(s)$ is the disjoint union of $X(s)$ and $Y(s)$ for s in A^* . The corresponding transition system has two starting states, contrary to the standard definition of transition systems—and so it may seem reasonable to restrict attention to rooted presheaves. In terms of category theory, that would mean working with *sheaves* instead of presheaves. Alternatively, we can simply view X and Y as presheaves over A^* ; such presheaves are easily seen to be in bijective correspondence with rooted presheaves over A^* .

The following result will be important later:

Proposition 4.4 *Suppose \mathbb{P} has an initial object, \perp . Any presheaf X in $\hat{\mathbb{P}}$ has a decomposition as a sum of rooted presheaves*

$$X \cong \sum_{i \in X(\perp)} X_i .$$

Here, for $i \in X(\perp)$, the presheaf X_i in $\hat{\mathbb{P}}$ is, to within isomorphism, given as

$$X_i(p) = \{x \in X(p) \mid (X!_p)x = i\} .$$

—where $!_p$ is the unique map $\perp \rightarrow p$ in \mathbb{P} .

Example 4.5 For a presheaf X over A^* , this rooted component decomposition exhibits the forest X as a sum of trees. \square

5 Natural transformations

The presheaves over a small category \mathbb{P} are the objects of the functor category $\widehat{\mathbb{P}} = [\mathbb{P}^{\text{op}}, \mathbf{Set}]$, with arrows being natural transformations. Spelled out, a natural transformation $\alpha : X \rightarrow Y$ between presheaves X and Y over \mathbb{P} is a family $(\alpha_p)_{p \in \mathbb{P}}$ of functions $\alpha_p : X(p) \rightarrow Y(p)$, satisfying that for any arrow $e : p \rightarrow p'$ of \mathbb{P} , the naturality square

$$\begin{array}{ccc}
 p' & X(p') & \xrightarrow{\alpha_{p'}} Y(p') \\
 \uparrow e & \downarrow X(e) & \downarrow Y(e) \\
 p & X(p) & \xrightarrow{\alpha_p} Y(p)
 \end{array}$$

commutes. It is perhaps not entirely obvious what process concept this amounts to, so we replay it in terms of categories of elements. Since each α_p is a function $X(p) \rightarrow Y(p)$, α induces a map $\mathcal{E}(\alpha)$ between the objects of $\mathcal{E}(X)$ and $\mathcal{E}(Y)$, sending (p, x) to $(p, \alpha_p x)$. Now, naturality of α is simply the same as functoriality of $\mathcal{E}(\alpha)$: that it sends morphisms $(p, x) \xrightarrow{e} (p', x')$ of $\mathcal{E}(X)$ to morphisms $(p, \alpha_p x) \xrightarrow{e} (p', \alpha_{p'} x')$ of $\mathcal{E}(Y)$. So it seems that a natural transformation from X to Y shows how Y may “simulate” X .

Example 5.1 We can define a natural transformation $\alpha : X \rightarrow Y$ between the presheaves of Example 3.1:

$$\begin{array}{ccccccc}
 (ac, \mathbf{0}) & & (at, \mathbf{0}) & & & & \\
 \uparrow & & \uparrow & \xrightarrow{\alpha_{ac}} & & & \\
 (a, \mathbf{0}) & & (a, \mathbf{1}) & & (ac, \mathbf{0}) & & (at, \mathbf{0}) \\
 \swarrow & & \swarrow & \xrightarrow{\alpha_a} & \swarrow & & \swarrow \\
 (\epsilon, \mathbf{0}) & & & & (a, \mathbf{0}) & & \\
 & & & & \uparrow & & \\
 & & & & (\epsilon, \mathbf{0}) & &
 \end{array}$$

Here, $(p, x) \xrightarrow{\alpha_p} (p, y)$ means that $\alpha_p x = y$, or equivalently that $\mathcal{E}(\alpha)$ maps (p, x) to (p, y) . \square

Since α preserves transitions in the categories of elements, we can think of it as a *simulation relation* on transition systems. Moreover, its graph is a function; such relations are called *functional simulations*. But to obtain a categorical definition of bisimulation, we need a way of *reflecting* as well as preserving behaviour.

6 Open maps

From a process viewpoint, it would be reasonable to make the extra requirement on a natural transformation $\alpha : X \rightarrow Y$ that whenever we have the situation on the left below, we may “complete the square” as on the right:

$$\begin{array}{ccc}
 (p', y') & & (p', x') \xrightarrow{\alpha_{p'}} (p', y') \\
 \uparrow e & & \uparrow e \\
 (p, x) \xrightarrow{\alpha_p} (p, y) & & (p, x) \xrightarrow{\alpha_p} (p, y)
 \end{array}$$

This can be formulated as the requirement that each naturality square of α (see opposite column) is a *quasi-pullback*, so that whenever we have $x \in X(p)$ and $y' \in Y(p')$ with $\alpha_p x = (Ye)y'$, then there exists an $x' \in X(p')$ such that $(Xe)x' = x$ and $\alpha_{p'} x' = y'$. In this case, we call α an *open map* of presheaves, following Joyal and Moerdijk [11].

Open maps between rooted presheaves, so trees, over A^* are functional bisimulations relating the roots of the trees. Since bisimilarity is an equivalence relation, the trees X and Y over A^* are bisimilar iff there exists a *span* of open maps between rooted presheaves:

$$\begin{array}{ccc}
 & Z & \\
 \alpha \swarrow & & \searrow \beta \\
 X & & Y
 \end{array}$$

For general, not necessarily rooted, presheaves open maps can still be thought of as functional bisimulations. However, since there may be no roots to relate, the induced notion of bisimulation via spans may imply no correspondence between the presheaves. In fact, there is a trivial span relating any pair of presheaves X, Y since $\widehat{\mathbb{P}}$ has an initial object (the empty presheaf, with empty contribution at each $p \in \mathbb{P}$). Again, we could take this as an indication to work only with rooted presheaves, or we could look for a requirement on the spans. As for the latter, demanding that the legs of the span are *epi* seems reasonable: open maps between rooted presheaves over A^* are automatically epi, and viewing trees over A^* as presheaves over A^+ , this extra requirement is exactly what is needed to show that bisimilarity as defined by open maps coincide with bisimilarity as defined by Park and

Milner. To avoid restricting the class of presheaves, we take

Definition 6.1 Presheaves X, Y over \mathbb{P} are *open-map bisimilar* if they are related by a span of epi open maps.

7 Yoneda

To each presheaf category $\widehat{\mathbb{P}}$ is associated a canonical functor $\mathbb{P} \rightarrow \widehat{\mathbb{P}}$, saying how to view paths as processes:

Definition 7.1 Let \mathbb{P} be a small category. The *Yoneda functor* $y_{\mathbb{P}} : \mathbb{P} \rightarrow \widehat{\mathbb{P}}$ maps $p \in \mathbb{P}$ to the presheaf $\mathbb{P}(-, p)$ and maps $e : p \rightarrow p'$ of \mathbb{P} to the natural transformation $\mathbb{P}(-, e)$ obtained by post-composition with e .

Example 7.2 The presheaf $A^*(-, s)$ is given by

$$A^*(-, s)(s') = A^*(s', s) = \begin{cases} \mathbf{1} & \text{if } s' \leq s \\ \mathbf{0} & \text{otherwise} \end{cases}$$

So $A^*(-, s)$ is a single branch with shape s , whereas $A^*(-, s \leq s')$ shows how it is included in a longer branch, with shape s' . \square

Intuitively, $y_{\mathbb{P}}$ maps a computation path shape p to a process that may do a single computation of shape p , and a path extension $e : p \rightarrow p'$ to a simulation of this computation by a longer one.

Following this intuition, a natural transformation $y_{\mathbb{P}}p \rightarrow X$ shows how the process X may simulate a process capable of performing just the path p . But then the set $X(p)$ of computation paths of X of shape p should be the same as the set of such natural transformations. This is the content of

Lemma 7.3 (Yoneda) *For any $p \in \mathbb{P}$ and $X \in \widehat{\mathbb{P}}$ there is a bijection $\widehat{\mathbb{P}}(y_{\mathbb{P}}p, X) \cong X(p)$.*

Example 7.4 For the presheaf X of Example 3.1 there are two elements $\mathbf{0}$ and $\mathbf{1}$ of $X(a)$ and so two natural transformations $A^*(-, a) \rightarrow X$. \square

Justified by the Yoneda lemma, if $(p, x) \in \mathcal{E}(X)$ we write x also for the corresponding natural transformation $y_{\mathbb{P}}p \rightarrow X$. This gives the following alternative characterisation of open maps:

Proposition 7.5 *A map $\alpha : X \rightarrow Y$ in $\widehat{\mathbb{P}}$ is open iff for every arrow $e : p \rightarrow p'$ of \mathbb{P} , a commuting square (on the left below) can be split into two commuting triangles (on the right):*

$$\begin{array}{ccc} y_{\mathbb{P}}p' & \xrightarrow{y'} & Y \\ y_{\mathbb{P}}e \uparrow & & \uparrow \alpha \\ y_{\mathbb{P}}p & \xrightarrow{x} & X \end{array} \quad \begin{array}{ccc} y_{\mathbb{P}}p' & \xrightarrow{y'} & Y \\ & \searrow x' & \uparrow \alpha \\ y_{\mathbb{P}}p & \xrightarrow{x} & X \end{array}$$

This characterisation of open maps highlights the possibility of replacing the Yoneda embedding with some other “inclusion” to obtain other notions of open maps and open map bisimulation. We return to this in Section 14.

8 Completeness

A presheaf category has all limits and colimits, given pointwise from the limits and colimits in **Set**. In particular, a presheaf category has all sums (co-products) of presheaves; the sum $\Sigma_{i \in I} X_i$ of presheaves X_i over \mathbb{P} has a contribution $\Sigma_{i \in I} X_i(p)$, the disjoint union of sets, at $p \in \mathbb{P}$. The empty sum of presheaves is the presheaf \emptyset with empty contribution at each $p \in \mathbb{P}$. In process terms, a sum of presheaves represents a nondeterministic sum of processes.

Example 8.1 Consider once again the two processes of Figure 2. If we view them as presheaves over A^+ , their nondeterministic sum, according to the above, corresponds to their sum as CCS terms. On the other hand, if we form their sum as rooted presheaves over A^* , we get the presheaf Z described in Section 4, which is not rooted, and cannot be expressed in CCS.

Similarly, the presheaf $\emptyset \in A^+$ corresponds to the inactive CCS process, also written \emptyset , while this is not so for $\emptyset \in A^*$ which intuitively cannot even do the empty string of actions. \square

As promised in Section 3, we can exhibit a process as a colimit of its paths:

Proposition 8.2 (Density formula) *Let X be a presheaf over \mathbb{P} . Then $X \cong \int^{(p,x) \in \mathcal{E}(X)} y_{\mathbb{P}}p$.*

The role of the colimit is to “glue together” the paths of X as dictated by the category of elements of X .

In the beginning of Section 7, we said that the Yoneda functor is “canonical”. We now make this precise: The functor $y_{\mathbb{P}} : \mathbb{P} \rightarrow \widehat{\mathbb{P}}$ satisfies the universal property that for any functor $F : \mathbb{P} \rightarrow \mathcal{C}$, where \mathcal{C} is a category with all colimits, there is a colimit-preserving functor $G : \widehat{\mathbb{P}} \rightarrow \mathcal{C}$, determined to within isomorphism, such that $F \cong G \circ y_{\mathbb{P}}$ —see *e.g.* [13], page 43:

$$\begin{array}{ccc} \mathbb{P} & \xrightarrow{y_{\mathbb{P}}} & \widehat{\mathbb{P}} \\ & \searrow F & \downarrow G \\ & & \mathcal{C} \end{array}$$

The proof uses the density formula. For $X \in \widehat{\mathbb{P}}$, we take $G(X)$ to be the colimit $\int^{(p,x) \in \mathcal{E}(X)} Fp$ in \mathcal{C} .

By taking \mathcal{C} to be a presheaf category $\widehat{\mathbb{Q}}$, this tells us that colimit-preserving functors between presheaf categories may be useful.

9 The category \mathbf{Lin}

Define the category \mathbf{Lin} to consist of small categories $\mathbb{P}, \mathbb{Q}, \dots$ with maps $G : \mathbb{P} \rightarrow \mathbb{Q}$ the colimit-preserving functors $\widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$. By the universal property, colimit-preserving functors $G : \widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$ correspond to within isomorphism to functors $F : \mathbb{P} \rightarrow \widehat{\mathbb{Q}}$, and such functors are in bijective correspondence with *profunctors*

$$\bar{F} : \mathbb{P} \leftrightarrow \mathbb{Q} .$$

Recall that the category of profunctors from \mathbb{P} to \mathbb{Q} , written $\mathbf{Prof}(\mathbb{P}, \mathbb{Q})$, is the functor category $[\mathbb{P} \times \mathbb{Q}^{\text{op}}, \mathbf{Set}]$, which clearly equals the category of presheaves $\widehat{\mathbb{P}^{\text{op}} \times \mathbb{Q}}$, and is isomorphic to the functor category $[\mathbb{P}, \widehat{\mathbb{Q}}]$. We thus have the chain of equivalences:

$$\mathbf{Lin}(\mathbb{P}, \mathbb{Q}) \simeq [\mathbb{P}, \widehat{\mathbb{Q}}] \cong \mathbf{Prof}(\mathbb{P}, \mathbb{Q}) = \widehat{\mathbb{P}^{\text{op}} \times \mathbb{Q}} .$$

The more symmetric, relational presentation via profunctors exposes an involution central in understanding \mathbf{Lin} as a categorical model of classical linear logic. The involution of linear logic, \mathbb{P}^{\perp} , on an object \mathbb{P} , is given by \mathbb{P}^{op} ; clearly presheaves over $\mathbb{P}^{\text{op}} \times \mathbb{Q}$ correspond to presheaves over

$(\mathbb{Q}^{\text{op}})^{\text{op}} \times \mathbb{P}^{\text{op}}$, showing how maps $G : \mathbb{P} \rightarrow \mathbb{Q}$ correspond to maps $G^{\perp} : \mathbb{Q}^{\text{op}} \rightarrow \mathbb{P}^{\text{op}}$ in \mathbf{Lin} .

The model is somewhat degenerate:

- The “par” $\mathbb{P} \wp \mathbb{Q}$ and tensor product $\mathbb{P} \otimes \mathbb{Q}$ of \mathbb{P} and \mathbb{Q} are both given by the product of small categories $\mathbb{P} \times \mathbb{Q}$. The neutral element is the terminal category $\mathbb{1}$ with one object and one arrow.
- Consequently, the function space from \mathbb{P} to \mathbb{Q} is given by $\mathbb{P}^{\text{op}} \times \mathbb{Q}$.
- On objects \mathbb{P} and \mathbb{Q} , products $\mathbb{P} \& \mathbb{Q}$ and coproducts $\mathbb{P} + \mathbb{Q}$ are both given by the disjoint juxtaposition of \mathbb{P} and \mathbb{Q} . The neutral element is given by the initial category $\mathbb{0}$ with no objects.
- As for the exponential $!$ of linear logic, there is room for choice [14], one of which is taken in Section 11.

10 A linear process language?

Consider interpreting a process language in the category \mathbf{Lin} . The rich type discipline seems promising, and furthermore, we have a congruence result “for free” in that

Proposition 10.1 [3] *Let $G : \widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$ be any colimit-preserving functor between presheaf categories. Then G preserves open maps and open-map bisimulation.*

So any process operation interpreted as a map of \mathbf{Lin} will automatically respect open-map bisimulation.

Unfortunately, because the colimit of the empty diagram is \emptyset , the maps of \mathbf{Lin} , unlike many process operations, always send the inactive process to the inactive process! In particular, we cannot interpret prefixing or parallel composition a la CCS in \mathbf{Lin} .

As a response, we can extend to maps from $!\mathbb{P}$ to \mathbb{Q} , for objects \mathbb{P} and \mathbb{Q} in \mathbf{Lin} . By the properties of the exponential, this allows arbitrary copying of the argument process, and so in particular allows a process operation to ignore its input in giving nontrivial output.

11 Copying

Intuitively, an object of $!\mathbb{P}$ should represent a computation path of an “assembly” of processes each with computation paths in \mathbb{P} —the assembly of processes can then be the collection of copies of a process, possibly at different states.

If we take $!\mathbb{P}$ to be the *finite-colimit completion* of \mathbb{P} , an object of $!\mathbb{P}$ as a finite colimit would express how paths coincide initially and then branch. One way to understand this object as a computation path of an assembly of processes is that the assembly is not fixed once and for all. Rather the assembly grows as further copies are invoked, and these copies can be made of processes after they have run for a while. The copies can then themselves be run and the resulting processes copied. In this way, by keeping track of the origins of copies, we can account for the identifications of subpaths.

With this choice of exponential, there is an obvious inclusion functor $i_{\mathbb{P}} : \mathbb{P} \rightarrow \widehat{\mathbb{P}}$, playing the role of the Yoneda embedding.

In particular, it can be shown that $\widehat{\mathbb{P}}$ with $i_{\mathbb{P}}$ is the free *filtered colimit* completion of $!\mathbb{P}$. Spelled out, given any functor $F : \mathbb{P} \rightarrow \mathcal{C}$ where \mathcal{C} has all filtered colimits, there is a filtered-colimit preserving functor $G : \widehat{\mathbb{P}} \rightarrow \mathcal{C}$, determined to within isomorphism, such that $G \circ i_{\mathbb{P}} \cong F$ (see [12]):

$$\begin{array}{ccc} !\mathbb{P} & \xrightarrow{i_{\mathbb{P}}} & \widehat{\mathbb{P}} \\ & \searrow F & \downarrow G \\ & & \mathcal{C} \end{array}$$

It follows that maps $!\mathbb{P} \rightarrow \mathbb{Q}$ in **Lin** correspond, to within isomorphism, to *continuous* (i.e., filtered colimit preserving) functors $\widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$.

12 The category Cts

Define **Cts** to be the category consisting of small categories $\mathbb{P}, \mathbb{Q}, \dots$ as objects and morphisms $F : \mathbb{P} \rightarrow \mathbb{Q}$ the continuous functors $F : \widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$; they compose as functors. Clearly **Lin** is a subcategory of **Cts**, one which shares the same objects. We have

$$\mathbf{Cts}(\mathbb{P}, \mathbb{Q}) \simeq \mathbf{Lin}(!\mathbb{P}, \mathbb{Q})$$

for all small categories \mathbb{P}, \mathbb{Q} . The category **Cts** is the coKleisli category of the comonad based on

finite-colimit completions. The unit of the corresponding adjunction is given by maps $copy : \mathbb{P} \rightarrow !\mathbb{P}$ of **Cts**, used to define prefixing below. We can easily characterise those maps in **Cts** which are in **Lin**:

Proposition 12.1 *Suppose $F : \widehat{\mathbb{P}} \rightarrow \widehat{\mathbb{Q}}$ is a functor which preserves filtered colimits. Then, F preserves all colimits iff F preserves finite coproducts.*

So a continuous map is linear iff it preserves sums.

There is an isomorphism $!(\mathbb{P} \& \mathbb{Q}) \cong !\mathbb{P} \times !\mathbb{Q}$ making **Cts** cartesian closed; this immediately allows us to interpret the simply-typed lambda calculus with pairing in **Cts**. Products $\mathbb{P} \& \mathbb{Q}$ in **Cts** are given as in **Lin** but now viewing the projections as continuous functors. The function space $\mathbb{P} \rightarrow \mathbb{Q}$ is given by $(!\mathbb{P})^{\text{op}} \times \mathbb{Q}$.

The category **Cts** does not have coproducts. However, we can build a useful sum in **Cts** with the help of the coproduct of **Lin** and $!$. Let $(\mathbb{P}_{\alpha})_{\alpha \in A}$ be a family of small categories. Their *prefixed sum*,

$$\Sigma_{\alpha \in A} \alpha. \mathbb{P}_{\alpha} ,$$

is based on the coproduct in **Lin** which is given by $\Sigma_{\alpha \in A} !\mathbb{P}_{\alpha}$ with corresponding injections $in_{\beta} : !\mathbb{P}_{\beta} \rightarrow \Sigma_{\alpha \in A} \alpha. \mathbb{P}_{\alpha}$, for $\beta \in A$. The *injections*

$$\beta.(-) : \mathbb{P}_{\beta} \rightarrow \Sigma_{\alpha \in A} \alpha. \mathbb{P}_{\alpha}$$

in **Cts**, for $\beta \in A$, are defined to be the compositions $\beta.(-) = in_{\beta} \circ copy$. As the notation suggests, $\beta.(-)$ is used to interpret prefixing with β .

The construction above satisfies a property analogous to the universal property of a coproduct. Suppose $F_{\alpha} : \mathbb{P}_{\alpha} \rightarrow \mathbb{Q}$ are maps in **Cts** for all $\alpha \in A$. Then, *within Lin*, we find a mediating map $F : \Sigma_{\alpha \in A} \alpha. \mathbb{P}_{\alpha} \rightarrow \mathbb{Q}$ determined to within isomorphism such that $F \circ \alpha.(-) \cong F_{\alpha}$ for all $\alpha \in A$. Since **Lin** is a subcategory of **Cts**, the mediating map F also belongs to **Cts**, but here it is not uniquely determined, not even up to isomorphism. Therefore, the prefixed sum is not a coproduct in **Cts**, but the linearity of the mediating map is just what we need for interpreting prefix match. Consider a prefix match term $[u > \beta.x \Rightarrow t]$ where t denotes a map $F_{\beta} : \mathbb{P}_{\beta} \rightarrow \mathbb{Q}$. We interpret it as the mediating map obtained for F_{β} together with constantly \emptyset maps $F_{\alpha} : \mathbb{P}_{\alpha} \rightarrow \mathbb{Q}$ for $\alpha \in A$ different from β , applied to the denotation of u .

13 Equivalence

The category $!\mathbb{P}$ has an initial element \perp , given by the empty colimit, and so we can decompose any presheaf X over $!\mathbb{P}$ as a sum of rooted presheaves. This is the key to a correspondence with the operational semantics of HOPLA. We may interpret $\mathbb{P} : a : \mathbb{P}'$ as a map

$$\mathbb{P} \rightarrow !\mathbb{P}'$$

of **Lin** by letting the judgement for β

$$\frac{\beta \in A}{\Sigma_{\alpha \in A} \alpha. \mathbb{P}_\alpha : \beta : \mathbb{P}_\beta}$$

denote restriction to $!\mathbb{P}_\beta$, and inductively interpreting $u \mapsto a$, $(a, -)$, $(-, a)$ as the denotation of a precomposed with the map given by application to $\llbracket u \rrbracket$, and first and second projections, respectively.

The rules are then sound in the sense that if $t \xrightarrow{a} t'$ then (identifying a term with its denotation) *copy* t' is a rooted component of $a(t)$, and in fact, there is a 1-1 correspondence between derivations with conclusion $t \xrightarrow{a} t'$, for some t' , and the rooted components of $a(t)$, so the rules are also complete.

14 Bisimulation

Interestingly, although the operational bisimulation of Park/Milner is a congruence for HOPLA, maps of **Cts** do not in general preserve open maps. This suggests that one should look at other notions of open map, *cf.* Section 7. Indeed, for each “inclusion”

$$i_{\mathbb{P}} : !\mathbb{P} \rightarrow \widehat{\mathbb{P}},$$

there is a corresponding notion of $i_{\mathbb{P}}$ -bisimulation. Unfortunately for our choice of exponential, $i_{\mathbb{P}}$ -bisimulation degenerates into isomorphism, but there are choices of exponential whose corresponding $i_{\mathbb{P}}$ -bisimulation coincide with open-map bisimulation, and one may hope to recover operational bisimulation as $i_{\mathbb{P}}$ -bisimulation for some choice of exponential (or expose it as a union of such bisimulations).

In fact, it appears that the correspondence between the denotational and operational semantics can be proved abstractly, depending only on properties of the prefixed sum, so that it holds also for other choices of exponential.

References

- [1] G. L. Cattani. *Presheaf Models for Concurrency*. BRICS Dissertation Series DS-99-1, 1999.
- [2] G. L. Cattani, M. P. Fiore, and G. Winskel. A theory of recursive domains with applications to concurrency. In *Proc. LICS'98*.
- [3] G. L. Cattani, A. J. Power, and G. Winskel. A categorical axiomatization for bisimulation. In *Proc. of CONCUR'98*, LNCS 1466, 1998.
- [4] G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the π -calculus. In *Proc. CTCS'97*, LNCS 1290.
- [5] M. P. Fiore, G. L. Cattani, and G. Winskel. Weak bisimulation and open maps. In *Proc. LICS'99*.
- [6] T. T. Hildebrandt. A fully abstract presheaf semantics for SCCS with finite delay. In *Proc. CTCS'99*, ENTCS 29.
- [7] T. T. Hildebrandt. *Categorical Models for Concurrency: Independence, Fairness and Dataflow*. BRICS Dissertation Series DS-00-1, 2000.
- [8] T. T. Hildebrandt, P. Panangaden, and G. Winskel. Relational semantics of non-deterministic dataflow. In *Proc. CONCUR'98*, LNCS 1466.
- [9] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21:666–677, 1978.
- [10] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127:164–185, 1996.
- [11] A. Joyal and I. Moerdijk. A completeness theorem for open maps. *Annals of Pure and Applied Logic*, 70:51–86, 1994.
- [12] G. M. Kelly. *Basic concepts of enriched category theory*. London Math. Soc. Lecture Note Series 64, CUP, 1982.
- [13] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. Springer-Verlag, 1992.
- [14] M. Nygaard and G. Winskel. Linearity in process languages. In *Proc. LICS'02*.
- [15] M. Nygaard and G. Winskel. A higher-order process language. In *Proc. CONCUR'02*.
- [16] R. Milner. *A Calculus of Communicating Systems*. LNCS 92, 1980.
- [17] G. Winskel. A presheaf semantics of value-passing processes. In *Proc. CONCUR'96*, LNCS 1119.